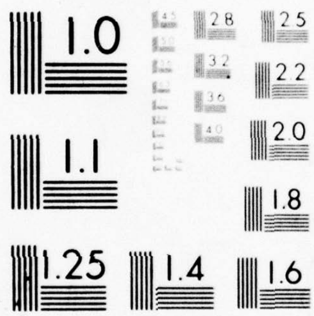
AD-A034 875 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 12/1
CLOSED-LOOP CONTROLS FOR DIFFERENTIAL GAMES USING A GRADIENT AN--ETC(U)
DEC 76 R R BACON
UNCLASSIFIED GA/MC/76D-2 NL

1 OF 1
AD
A034875

END
DATE
FILMED
2-77



MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS 1963

D D C
JAN 26 1977

1

CLOSED-LOOP CONTROLS FOR DIFFERENTIAL GAMES USING A GRADIENT AND A DIFFERENTIAL DYNAMIC PROGRAMMING METHOD

THESIS

GA/MC/76D-2

Robert R. Bacon
Captain USAF

D D C
JAN 26 1977

Approved for public release; distribution unlimited

(14) GA/MC/76D-2 (6)

CLOSED-LOOP CONTROLS FOR DIFFERENTIAL GAMES USING A GRADIENT AND A DIFFERENTIAL DYNAMIC PROGRAMMING METHOD.

THESIS (9) Master's thesis,

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

ACCESSION for
NTIS White Section
DDC Buff Section
UNANNOUNCED
JUSTIFICATION
BY
DISTRIBUTION/AVAILABILITY CODES
Dist. AVAIL. and/or SPECIAL
A

by
(10) Robert R. Bacon, B.S.As.E. (12) 83p.
Captain USAF
Graduate Astronautical Engineering

(11) December 1976

Approved for public release; distribution unlimited.

1473
012225
LB

Preface

This thesis investigates the application of a gradient method and a combined gradient-differential dynamic programming method to generate a closed-loop control law for intercept problems formulated as differential games. The two example problems presented in this report are somewhat simplified and serve mainly to test the methods. It is expected that the general results will carry over in a satisfactory manner to problems with more complicated dynamics.

My sincere appreciation goes to Prof. Gerald M. Anderson for his advice and guidance in this effort and to my wife, Lynn, for her encouragement and endurance of many lonely hours during the preparation of this work.

This thesis is dedicated to Capt. Arthur J. Brandt whose tragic death grieved all of us in the GA-76D class, to my wife Lynn, and to my son Kyle.

Contents

List of Figures

List of Tables

List of Symbols

Symbol	Definition
A	Predicted Cost Change
E	Evader (Maximizing Player)
H	Hamiltonian function
J	Cost functional
K_E	Weighting Matrix (Evader)
K_P	Weighting Matrix (Pursuer)
$KOUNT_{MAX}$	Maximum allowable number of iterations
L	Path dependant portion of cost
P	Pursuer (Minimizing Player)
T	Target on Earth's surface
T_E	Evader's constant thrust
T_P	Pursuer's constant thrust
t_f	Terminal (final) time
t_o	Initial time
t_s	Sampling time
u	Minimizing player's control vector
v	Maximizing player's control vector
x	State vector
$(\dot{\ })$	Differentiation of ( ) with respect to time
$\|(\ )\|$	The Euclidean norm of ( )

$(\ \ )\big\|_{0}$	Evaluation of ( ) along a nominal path
$(\ \ ) \rightarrow (\ \ )$	( ) goes to ( )
$\delta(\ \ )$	Small variation in ( )
Δ	Sampling interval
ϵ	A small number
λ	Costate vector
ϕ	Terminal cost
τ	Time-to-go

<u>Subscripts</u>

E	Denotes Evader (maximizing player)
f	Terminal value
I	Inertial coordinate axis fixed on Earth's surface
i	The i^{th} component
NONOPT	Denotes non-optimality
o	Denotes nominal
P	Denotes Pursuer (minimizing player)
u	Partial differentiation with respect to u
v	Partial differentiation with respect to v
x	Partial differentiation with respect to x

<u>Superscript</u>

$*$	Denotes optimality
T	Transpose operation

Abstract

This thesis investigates the use of a gradient method and a combined gradient-differential dynamic programming (DDP) method to generate closed-loop controls for intercept problems formulated as differential games. The gradient method is applied to a planar motion pursuit-evasion game. The trajectory obtained compares favorably with that obtained using analytic expressions for closed-loop controls. The gradient method is applied to a three dimensional interceptor-penetrator game with simplified dynamics on a real-time basis. A combined gradient-DDP algorithm is applied to this problem but not on a real-time basis. The DDP portion of this combined control law was found to be unstable. The results obtained indicate that a gradient based scheme, because of its numerical stability and ability to rapidly converge to the vicinity of the optimum, may be used to generate an effective near optimal closed-loop control law for some problems.

CLOSED-LOOP CONTROLS FOR DIFFERENTIAL GAMES USING A GRADIENT AND A DIFFERENTIAL DYNAMIC PROGRAMMING METHOD

I. Introduction

Background

Differential game theory can be applied to many problems where two participants have diametrically opposing goals. Many problems encountered in the area of military strategy can be modeled as a two-person zero-sum game.

In a zero-sum game, one player applies a control in an effort to minimize a cost (performance index) and the other player applies a control to maximize the cost. Application of necessary conditions for a saddle point solution to the game will yield a two-point boundary value problem (TPBVP) which must be solved to generate the optimal controls for both players. Solution of this TPBVP for a given set of state initial conditions will yield open-loop strategies. Strategies, or control laws, so obtained are useful if it can be assumed that both players will use their optimal controls throughout the duration of the game. Unfortunately, for most realistic problems this is an invalid assumption.

In order to use differential game theory in problems of aerial combat and interception, a closed-loop control law is needed so that one player may immediately capitalize

upon non-optimal play by his opponent. A computational algorithm is needed to provide a closed-loop control law.

If one of the players, for example E (Evader), uses a non-optimal control, then the other player, P (Pursuer), updates his control vector by use of a computational algorithm. To accomplish this task, P samples the system state at fixed time intervals and computes new controls using the sampled state vector and computed (or assumed) nominal control histories for both players. Between sampling times, P flies open-loop using controls computed at the previous sampling time.

Generation of a near-optimal closed-loop control using neighboring extremal algorithms has been accomplished with some success (Refs 1,2). A major drawback to using these methods is that they require the use of a reference trajectory. In order to obtain a reference trajectory, a TPBVP must be solved and, typically, this requires a large amount of computational time. In addition, numerical stability of neighboring extremal methods sometimes causes convergence problems.

An ideal closed-loop control law would be one that is stable, depends only on the current state, requires no reference trajectory, provides an optimal or near-optimal control law, and can be implemented on a real-time basis.

Statement of the Problem

Scope. This thesis will investigate the use of a numerical algorithm based upon a gradient scheme that meets the above criteria and may be used to generate a near-optimal closed-loop control law. Two example problems are presented to test this method. The first problem is a planar relative motion pursuit-evasion game and the second is a three dimensional interceptor-penetrator game.

Approach. In the planar pursuit-evasion game, the trajectory obtained using the gradient closed-loop control law will be compared with the trajectory obtained using analytic expressions for the closed-loop controls. The gradient method will be applied to the interceptor-penetrator game on a real-time basis. Effects of varying convergence parameters and state sampling interval will be studied. A differential dynamic programming (DDP) method is used in conjunction with the gradient method in an effort to generate a more accurate closed-loop control law for the interceptor-penetrator problem.

II. Differential Game Formulation and Closed-Loop Control Laws

Many aerial combat and interception problems, including the two problems considered in this thesis, may be formulated as two-person zero-sum differential games. The non-linear system dynamics of the game are modeled using a first order vector differential equation,

$$\dot{x} = f(x, u, v) \tag{1}$$

with initial conditions

$$x(t_o) = x_o \tag{2}$$

The scalar cost functional is

$$J = \phi[x(t_f), t_f] + \int_{t_o}^{t_f} L(x, u, v)\, dt \tag{3}$$

where the vector, u, contains the controls of the minimizing player and the vector, v, contains the controls of the maximizing player.

The construction of a cost functional that incorporates all the necessary features of the conflict and does not bias the outcome toward one player is a problem in itself. The cost used in pursuit-evasion games normally includes some measure of the range between the two vehicles at the

end of the game. Formulation of a payoff for other types of encounters (e.g. interceptor-penetrator) within the context of zero-sum, two-person games is more difficult.

The optimal solution to a zero-sum differential game depends on the stationarity of Equation (3). Thus, the problem is to find the control pair (u^*, v^*) such that

$$J(u^*, v) \leq J(u^*, v^*) \leq J(u, v^*) \quad . \tag{4}$$

The middle term of (4) is known as the value of the game and an asterik (*) denotes the optimal nature of the control vector. This inequality is called a game theoretic saddle point and incorporates the idea that if one player uses a control other than the optimal, then the other may capitalize upon this to cause a change in the value of the game to his favor.

For a game with no terminal state constraints, the necessary conditions for a saddle point solution to the game are

$$\dot{x} = f(x, u, v) \quad , \quad x(t_0) = x_0 \tag{5}$$

$$\dot{\lambda} = -H_x \quad , \quad \lambda(t_f) = \phi_x[x(t_f), t_f] \tag{6}$$

$$H_u = 0 \tag{7}$$

$$H_{uu} \geq 0 \tag{8}$$

$$H_v = 0 \tag{9}$$

$$H_{vv} \leq 0 \tag{10}$$

$$H(t_f) = -\phi_t \tag{11}$$

where the Hamiltonian, H, is defined as

$$H(x,\lambda,u,v) = L(x,u,v) + \lambda^T f(x,u,v) \quad . \tag{12}$$

If there is no path cost, then L is zero in Equations (3) and (12). A derivation of these necessary conditions is given in Chapters 2 and 9 of Reference 3. The TPBVP that must be solved is given by Equations (5) and (6). Expressions for the optimal controls in terms of the state and adjoint variables are obtained from Equations (7) through (10). Equation (11) determines the time at which the game terminates and can be shown to be equivalent to the expression

$$\dot{J} = 0 \quad . \tag{13}$$

If the final time is prespecified, then Equation (11) may be omitted.

Solution of the TPBVP will yield open-loop control strategies, (u^*, v^*), which represent the "best" one player can do against the "best" of his opponent. If the opponent

chooses to use a control other than v^*, then the first player must update his control based on the current system state and time. This results in a closed-loop control law. A quality of two-person zero-sum games is that if both players use their optimal controls, then the open-loop and closed-loop trajectories are identical.

A Gradient Closed-Loop Control Law

Adjoining the system dynamics, Equation (1), to the scalar cost functional, Equation (3), by use of time varying LaGrange multipliers (adjoint variables), one has

$$J = \phi[x(t_f), t_f] + \int_{t_0}^{t_f} [L(x,u,v) + \lambda^T(f(x,u,v) - \dot{x})]\,dt \quad . \quad (14)$$

Substituting for the Hamiltonian, Equation (14) becomes

$$J = \phi[x(t_f), t_f] + \int_{t_0}^{t_f} [H(x,\lambda,u,v) - \lambda^T \dot{x}]\,dt \quad . \quad (15)$$

Along nominal state and adjoint variable trajectories, x and λ, a variation in the controls from the nominal controls u_o , v_o results in

$$J + \delta J = \phi[x(t_f), t_f] + \int_{t_0}^{t_f} [H(x,\lambda,u_0+\delta u, v_0+\delta v) - \lambda^T \dot{x}]\,dt \quad . \quad (16)$$

Expansion of Equation (16) in a Taylor's series through first order terms about the nominal values yields

$$\delta J = \int_{t_0}^{t_f} [H_u\big|_0 \delta u + H_v\big|_0 \delta v] \, dt \qquad , \qquad (17)$$

which is the first order variation in the cost due to small control changes δu and δv (Ref 4). The adjoint variables used in Equation (17) must satisfy the expressions given by Equations (6).

In a differential game, player P applies a control change, δu, in an attempt to minimize the cost. Player E is assumed to be applying a control change, δv, to maximize the cost. Therefore, in Equation (17), the i^{th} component of the vector, δu, is adjusted so as to be of opposite sign of the i^{th} component of the vector, H_u. Likewise, the i^{th} component of the vector, δv, is adjusted so as to be of the same sign as the i^{th} component of the vector, H_v. This procedure ensures that the opposing players are making control changes so as to cause a change in the cost, δJ, to their favor.

To use the gradient method to generate a closed-loop control law, the control corrections, δu and δv, are computed at discrete sampling times. At each sampling time, t_s, the gradient closed-loop control algorithm will be called upon by player P to generate an updated control based upon the

sampled state vector. The cost variance over the time interval, $t_f - t_s$, (the predicted time-to-go) is given by

$$\delta J = \int_{t_s}^{t_f} [H_u|_0 \delta u + H_v|_0 \delta v] \, dt \quad . \qquad (18)$$

The control corrections made on each iteration of the algorithm are computed with

$$\delta u = K_P H_u \qquad \text{and}$$

$$\delta v = K_E H_v \qquad \text{, where} \qquad (19)$$

K_P and K_E are diagonal weighting (step-size) matrices of proper dimension to make (19) conformable. K_P and K_E are selected so that $K_P < 0$ (negative definate) and $K_E > 0$ (positive definate).

<u>Selection of the Step-Size</u>. Gradient methods are inherently stable and converge to the general vicinity of a local optimal solution within a few iterations if the step-size matrices, K_E and K_P, are selected properly. Leatham (Ref 4) suggests using

$$(K_P)_{ii} = (\delta u_i)_{MAX} / (H_{u_i})_{MAX} \qquad \text{and}$$

$$(K_E)_{ii} = (\delta v_i)_{MAX} / (H_{v_i})_{MAX} \qquad , \qquad (20)$$

where $(\delta u_i)_{MAX}$ and $(\delta v_i)_{MAX}$ represent the largest control

corrections that the designer estimates could be applied on each iteration. $(H_{u_i})_{MAX}$ and $(H_{v_i})_{MAX}$ are the maximum control gradient magnitudes which are estimated values for the first iteration and updated, if necessary, on subsequent iterations. This provides a simple method for taking a variable step-size to aid convergence.

An alternate method of computing K_P and K_E arises by the inclusion of second order terms in Equation (17). In doing this, the cost change over the predicted time-to-go $(t_f - t_s)$ becomes

$$\delta J = \int_{t_s}^{t_f} \left[H_u\big|_o \delta u + \delta u^T H_{uu}\big|_o \delta u + H_v\big|_o \delta v + \delta v^T H_{vv}\big|_o \delta v \right] dt \quad . \tag{21}$$

For the variation δJ to vanish, it is necessary that

$$\left[H_u\big|_o + \delta u^T H_{uu}\big|_o \right] \delta u = 0 \quad ;$$

$$\left[H_v\big|_o + \delta v^T H_{vv}\big|_o \right] \delta v = 0 \quad . \tag{22}$$

From Equations (22) the control corrections are found to be

$$\delta u = \left[-H_{uu}^{-1} H_u \right]\Big|_o \quad ;$$

$$\delta v = \left[-H_{vv}^{-1} H_v \right]\Big|_o \tag{23}$$

provided H_{uu}^{-1} and H_{vv}^{-1} exist. In this case, the terms $-H_{uu}^{-1}$ and $-H_{vv}^{-1}$ are analogous to K_E and K_P in Equations (19).

Convergence of the Gradient Method. In this thesis, convergence of the gradient method is defined to occur when the square of the Euclidean norms of the gradient vectors H_u and H_v at the sampling time are both less than some small number. That is,

$$\|H_u(t_s)\|^2 \leq \epsilon \qquad \text{and}$$

$$\|H_v(t_s)\|^2 \leq \epsilon \qquad . \qquad (24)$$

Convergence can also be defined to occur when δJ in Equation (18) is small.

A Gradient Method Control Update Algorithm. For a free final time problem with no terminal state constraints, the algorithm used in this thesis is summarized as follows:

(1) From the current state and sampling time ($x(t_s)$ and t_s) integrate the state equations forward until $\dot{J} \rightarrow 0$. This establishes a predicted final state and time ($x(t_f)$ and t_f). Use the current nominal control histories u_o , v_o in this integration. The u_o and v_o used for the first iteration at the

first sample time must be chosen by some reasonable logic (e.g. proportional navigation or line of sight).

(2) Evaluate the adjoint variable boundary condition, $\lambda(t_f) = \phi_x[x(t_f), t_f]$. This is the same expression as given in Equation (6).

(3) Integrate the state and adjoint differential equations (see Equations (5) and (6)) back in time to t_s using the nominal controls u_o and v_o. Use $\lambda(t_f)$ from Step (2) and $x(t_f)$ from Step (1) as boundary conditions for this integration. Compute and store $H_u|_o$ and $H_v|_o$ (also $H_{uu}|_o$ and $H_{vv}|_o$ if needed) along this backward trajectory.

(4) Update the control histories $u_o = u_o + \delta u$; $v_o = v_o + \delta v$ where $\delta u = K_P H_u$ and $\delta v = K_E H_v$. The signs of $(\delta u)_i$ and $(\delta v)_i$ are adjusted so that $(\delta u)_i$ and $(H_u)_i$ are of opposite sign and $(\delta v)_i$ and $(H_v)_i$ are of the same sign. K_P and K_E are defined by Equations (20) or (23).

(5) Repeat Steps (1) through (4) above until $\| H_u(t_s)\|^2 \leq \epsilon$ and $\| H_v(t_s)\|^2 \leq \epsilon$. If this convergence criterion is not met, then computation is terminated after a prespecified maximum allowable number of iterations.

For the problems considered in this thesis, the predicted time-to-go $(t_f - t_s)$, divided by sixty-four, was used as the

integration step-size in the control update algorithm. This mechanization has the feature of using smaller and smaller integration steps as t_s approaches t_f. Because of this, higher accuracy of the numerical integration method can be expected as the game progresses.

A Differential Dynamic Programming Closed-Loop Control Law

The change in the cost for the time period $t_f - t_s$, due to small deviations in the controls from the nominal, is (Ref 5)

$$\delta J = \int_{t_s}^{t_f} [H(x, \lambda, u_o + \delta u, v_o + \delta v) - H(x, \lambda, u_o, v_o)]\, dt \quad . \quad (25)$$

By letting A = δ J and performing the indicated integration, one has the total predicted cost change, A, that would result by applying the controls $u_o + \delta u$ and $v_o + \delta v$. The equivalence of Equations (25) and (18) for small δu and δv may be shown by a Taylor's series expansion through first order terms of Equation (25).

For a separable Hamiltonian (one which may be separated into a portion containing P's controls and a portion containing E's controls), one has

$$A = A_E + A_P = \int_{t_s}^{t_f} [H_E(x, \lambda, v^*) + H_P(x, \lambda, u^*) -$$

$$H_E(x, \lambda, v_o) - H_P(x, \lambda, u_o)]\, dt \quad , \quad (26)$$

where $v^* = v_o + \delta v$ and $u^* = u_o + \delta u$. The predicted cost change due to δv is A_E and the predicted cost change due to δu is A_P. Convergence at each sampling time occurs when the total predicted cost change, A, is small and

$$\epsilon > A_E \geq 0 \quad ;$$

$$-\epsilon < A_P \leq 0 \quad . \tag{27}$$

Equations (27) indicate that E applies a control change to cause an increase in the cost and P applies a control change to cause a decrease in the cost.

<u>A DDP Method Control Update Algorithm.</u> For a free final time problem with no terminal state constraints, the algorithm is summarized as follows:

(1) From the current state and sampling time ($x(t_s)$ and t_s) integrate the state equations forward until $\dot{J} \rightarrow 0$. This establishes a predicted final state and time ($x(t_f)$ and t_f). Use the current nominal control histories, u_o and v_o, in this integration.

(2) Evaluate the adjoint variable boundary condition, $\lambda(t_f) = \phi_x[x(t_f), t_f]$. This is the same expression as given in Equation (6).

(3) Integrate the state and adjoint differential equations back in time to t_s using the nominal controls u_o,

v_o. Use $\lambda(t_f)$ from Step (2) and $x(t_f)$ from Step (1) as boundary conditions for this integration. Along this backward trajectory:

(a) Compute and store u^* and v^* which are found from the optimality conditions given by Equations (7) through (10). In many problems, analytic expressions for u^* and v^* in terms of the state and adjoint variables may be obtained.

(b) Compute H_{Eold} and H_{Pold} using the current state and adjoint variables and the nominal controls u_o, v_o. Compute H_{Enew} and H_{Pnew} using the current state and adjoint variables and the new controls u^*, v^*.

(c) Form the equations

$$\dot{A}_E = H_{Eold} - H_{Enew}\text{, and}$$
$$\dot{A}_P = H_{Pold} - H_{Pnew}\,.$$

$\dot{A}_E$ and $\dot{A}_P$ are integrated backwards with the state and adjoint differential equations from the boundary conditions $A_E(t_f) = A_P(t_f) = 0$.

(4) Update the nominal control histories by replacing u_o with u^* and v_o with v^*.

(5) Convergence occurs when the total predicted cost change at t_s, $A(t_s) = A_E(t_s) + A_P(t_s)$, is less than some small number, and when $\epsilon > A_E(t_s) \geq 0$ and $-\epsilon < A_P(t_s) \leq 0$.

Note that this DDP scheme incorporates no check on the size of the control corrections made on each iteration of the algorithm. In the application of this method to the problem of Chapter IV, it is assumed that the nominal controls initially provided to the DDP scheme are sufficiently close to the optimum to obviate the need for a convergence control method.

Game Simulation

Game computer simulations employing both the gradient and combined gradient-DDP control update algorithms use the following general set-up:

(1) Set P's sampling interval, Δ .

(2) Set E's non-optimal control sequence, v_{NONOPT}.

(3) Establish reasonable nominal control histories u_o , v_o for use in the control update algorithm on the first iteration at the first sampling time. A reasonable choice might be a control based upon line-of-sight or proportional navigation.

(4) Integrate the state differential equations forward from the given initial conditions, $x(t_o)$, using the controls v_{NONOPT} for E and u_o for P.

(5) When the state sampling time, t_s, is reached, P calls upon the control update algorithm (gradient or gradient-DDP in this thesis).

(6) After updating his control, P flies open-loop until the next sampling time.

Computer simulations of the differential games considered in this thesis were run on a Control Data Corporation 6600 digital computer. The integration routines used in the control update algorithms employ a fourth order Runge-Kutta method to start and then use a four point Adams-Bashforth-Moulton predictor corrector scheme.

Algorithm Implementation on a Real-Time Basis. Since the control update algorithm requires a small but finite amount of computational time, there is a time lag between the state sampling time and the time that P may actually implement the updated control. This may be pictured using a time line as shown in Figure 1. Note that if P chooses a sampling interval too small, this may prevent implementation of the updated control before the next sampling time is reached. This situation will decrease the effectiveness of the control updating scheme.

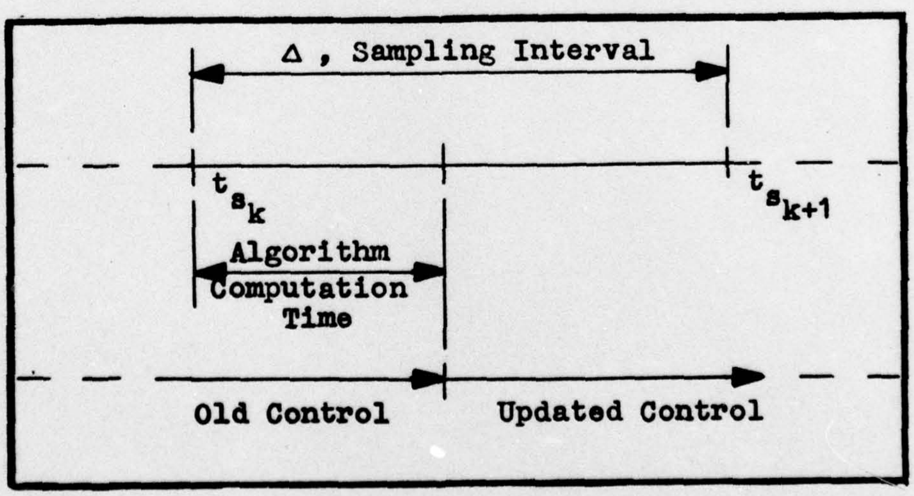


Figure 1. Control Implementation Time Line

III. Planar Relative Motion Pursuit-Evasion Game

Game Formulation

Consider a simple fixed end time planar relative motion pursuit-evasion game involving two constant thrust rockets. The geometry of the problem is shown in Figure 2. The origin of the XY cordinate system is centered at P and the equations of motion of E relative to P are,

$$\dot{X} = V_X$$

$$\dot{V}_X = T_E \cos v - T_P \cos u$$

$$\dot{Y} = V_Y$$

$$\dot{V}_Y = T_E \sin v - T_P \sin u \quad . \qquad (28)$$

The initial conditions for this problem are

$$X(0) = 10$$

$$V_X(0) = 0$$

$$Y(0) = 0$$

$$V_Y(0) = 0 \quad . \qquad (29)$$

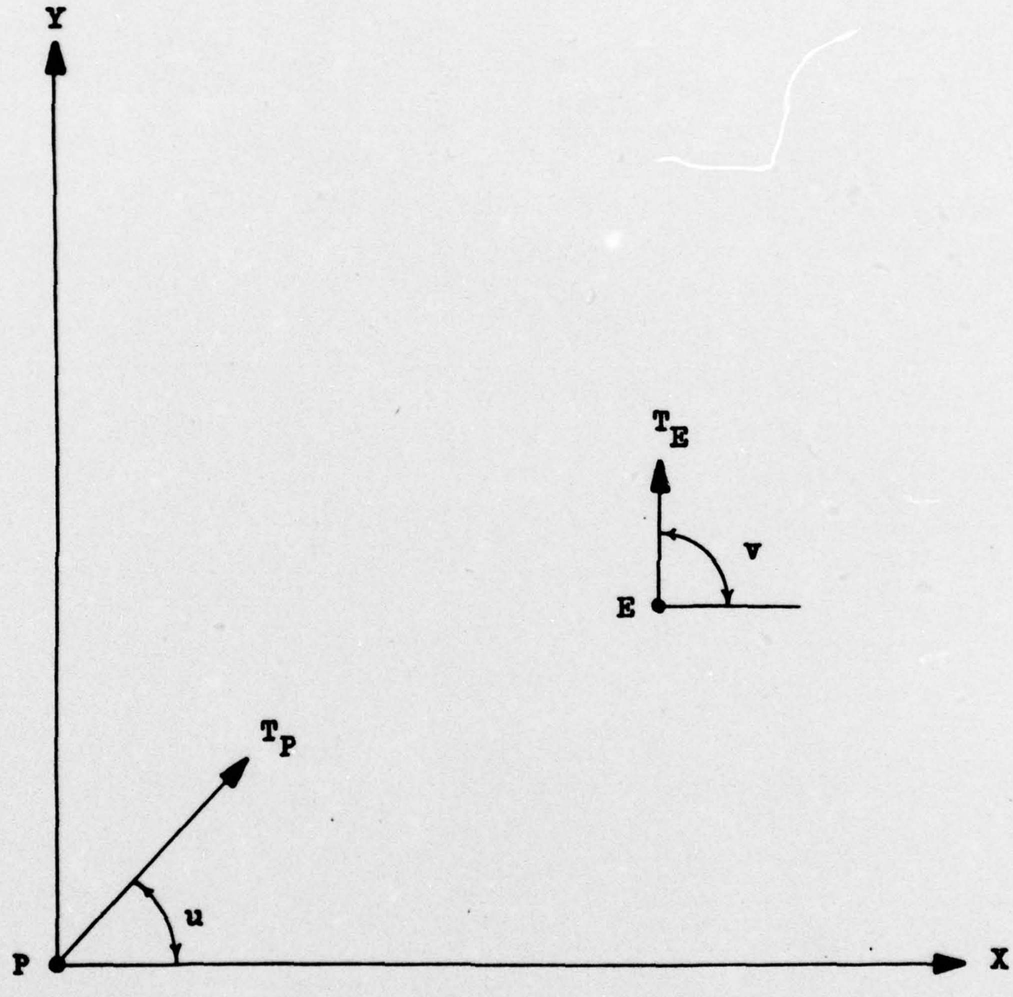


Figure 2. Planar Pursuit-Evasion Game Geometry

The payoff is taken to be one half the square of the final EP range,

$$J = \tfrac{1}{2}\left[X^2 + Y^2\right]\Big|_{t_f} \quad . \qquad (30)$$

In this problem, drag and gravity effects are neglected and it is assumed that both vehicles are point masses using constant thrust. The thrust levels used are $T_P = 1.0$ and $T_E = 0.5$.

Application of Necessary Conditions

The Hamiltonian for this problem is

$$\begin{aligned} H = {} & \lambda_x V_x + \lambda_{V_x}\left[T_E \cos v - T_P \cos u\right] + \\ & \lambda_y V_y + \lambda_{V_y}\left[T_E \sin v - T_P \sin u\right] \quad . \qquad (31) \end{aligned}$$

The adjoint differential equations and boundary conditions are given by Equations (6) and for this problem are found to be

$$\begin{aligned} \dot{\lambda}_x &= 0 \quad , & \lambda_x(t_f) &= X_f \\ \dot{\lambda}_{V_x} &= -\lambda_x \quad , & \lambda_{V_x}(t_f) &= 0 \\ \dot{\lambda}_y &= 0 \quad , & \lambda_y(t_f) &= Y_f \\ \dot{\lambda}_{V_y} &= -\lambda_y \quad , & \lambda_{V_y}(t_f) &= 0 \quad . \qquad (32) \end{aligned}$$

By applying the optimal control conditions given by Equations (7) through (10) it is found that

$$\sin u^* = \sin v^* = \frac{\lambda_{v_y}}{\sqrt{\lambda_{v_x}^2 + \lambda_{v_y}^2}}$$

$$\cos u^* = \cos v^* = \frac{\lambda_{v_x}}{\sqrt{\lambda_{v_x}^2 + \lambda_{v_y}^2}} \quad . \qquad (33)$$

Equations (33) constitute an open-loop strategy for this game. The TPBVP given by Equations (28), (29), (32), and (33) must be solved to generate the solution to the game.

With the initial conditions given by (29) the optimal strategy is

$$u^* = 0 \qquad \text{and} \qquad v^* = 0 \quad . \qquad (34)$$

If the evader (E) uses a non-optimal control $v_{NONOPT} = 90°$ then the pursuer (P) must update his control in order to take advantage of E's non-optimal play.

<u>Closed-Loop Controls</u>

<u>Analytic Closed-Loop Controls</u>. Analytic expressions for P's optimal closed-loop controls are

$$\sin u^* = \frac{\bar{Y}}{\sqrt{\bar{X}^2 + \bar{Y}^2}}$$

$$\cos u^* = \frac{\bar{X}}{\sqrt{\bar{X}^2 + \bar{Y}^2}} \quad , \qquad (35)$$

where $\bar{X} = V_x(t_f - t) + X$ and $\bar{Y} = V_Y(t_f - t) + Y$.

A derivation of this control law is provided in Appendix A.

The trajectory obtained using the closed-loop control law (35) will provide a basis for comparing the trajectory obtained by using a gradient closed-loop control law.

Computed Closed-Loop Controls. The gradient scheme mechanized to compute P's control correction, δu, is given by Equation (23). A fixed flight time of 4.5 time units was used in the computer simulation of this problem. The control update algorithm for this problem is not implemented in real-time.

The convergence criterion used was

$$|H_u(t_s)| \leq \epsilon \quad \text{and} \quad |H_v(t_s)| \leq \epsilon \quad .$$

If this convergence criterion was not met then computation was terminated after 25 iterations. The game was played using various sampling intervals (Δ) and convergence factors (ϵ).

Results

Analytic Solution vs. Computed (Gradient Method) Solution. The trajectory obtained using the gradient control update scheme compared very well with the trajectory found from the analytic expression for the closed-loop controls. The

trajectories are shown in Figure 3. A sampling interval of $\Delta = 0.1406$ time units and a convergence factor of $\epsilon = 0.1$ were used in the gradient method run.

Variation in Cost with Convergence Factor. The convergence factor, ϵ, is a qualitative indicator of how near the computed solution is to the optimal solution. The smaller the value of ϵ, the closer the solution will be to the optimum. If the gradient closed-loop control law is to be implemented on a real-time basis, then the number of iterations of the algorithm at each sampling time must be kept low to minimize the computational time. Some accuracy is sacrificed in this effort.

In order to study the degree of accuracy of the computed solution, the game was played using differing convergence factors. The results are given in Table 1. Note from Table 1 that the more stringent convergence criteria ($\epsilon = 0.01$ and $\epsilon = 0.001$) resulted in more iterations of the algorithm and did not significantly reduce the cost. This result indicates that a closed-loop gradient method would provide reasonably accurate solutions in a few iterations if the value of ϵ is not too small.

Variation in Cost with Sampling Interval. Using a convergence factor of $\epsilon = 0.1$ the variation in cost was studied for different sampling intervals. The results are shown in Table 2. The cost approaches the analytic

Figure 3. Planar Pursuit-Evasion Game Trajectories

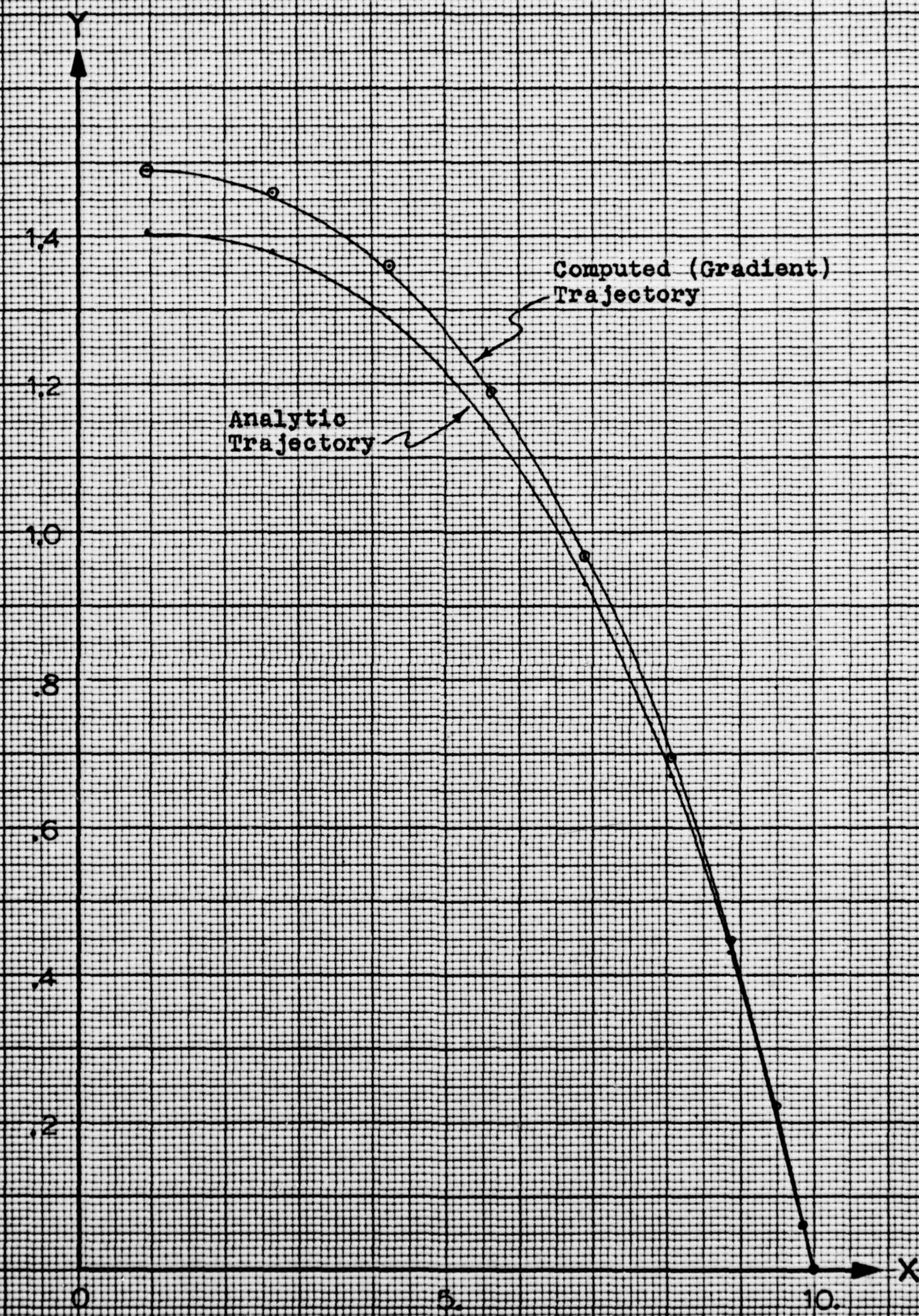


Table 1. Variance in Cost with Convergence Factor			
Guidance Scheme	Final Cost	Number of times convergence criteria not met	Maximum no. of iterations used at any sampling time
Analytic	1.45226	n/a	n/a
Gradient (ϵ = 0.1 , Δ = 0.14063)	1.56053	none	3
Gradient (ϵ = 0.01 , Δ = 0.14063)	1.54187	2	25
Gradient (ϵ = 0.001 , Δ = 0.14063)	1.54259	11	25

Table 2. Variance in Cost with Sampling Interval						
Sampling Interval (Δ)	1DT*	10DT	20DT	30DT	40DT	50DT
Cost	1.487	1.708	2.463	2.406	2.889	3.485

*DT = 4.5/128.

solution cost as the sampling interval decreases.

A real-time implementation of the gradient closed-loop control method is made in the problem of the next chapter.

IV. Three Dimensional Interceptor-Penetrator Game

Game Formulation

In an interceptor-penetrator encounter, the penetrator (designated with an E) attempts to penetrate the defenses of a fixed target and hit that target or at least maneuver so as to be in a favorable position to hit the target at some later time. The interceptor (designated with a P) attempts to intercept E or at least force E into an unfavorable position from which to hit T. This type of conflict may involve a bomber and a surface-to-air missile. The conflict could also be between a maneuverable intercontinental ballistic missile (ICBM) and an anti-ballistic missile (ABM) missile.

Consider a simplified dynamical model of an encounter between an ICBM (E) and an ABM (P). Capital letters designate E's state and lower case letters designate P's state. The geometry of this problem is shown in Figure 4. The constant thrust to mass ratios available to E and P are T_E and T_P respectively. The controls are the angles v_1 and v_2 for E and u_1 and u_2 for P. The first control angles (v_1 and u_1) are measured in the azimuth plane from the positive x_I axis and the second angles (v_2 and u_2) are measured from the local horizontal.

Scaling. In this problem the following scaling is used:

1 distance unit (d.u.) $\sim 10^4$ feet

1 time unit (t.u.) $\sim$ 1 second

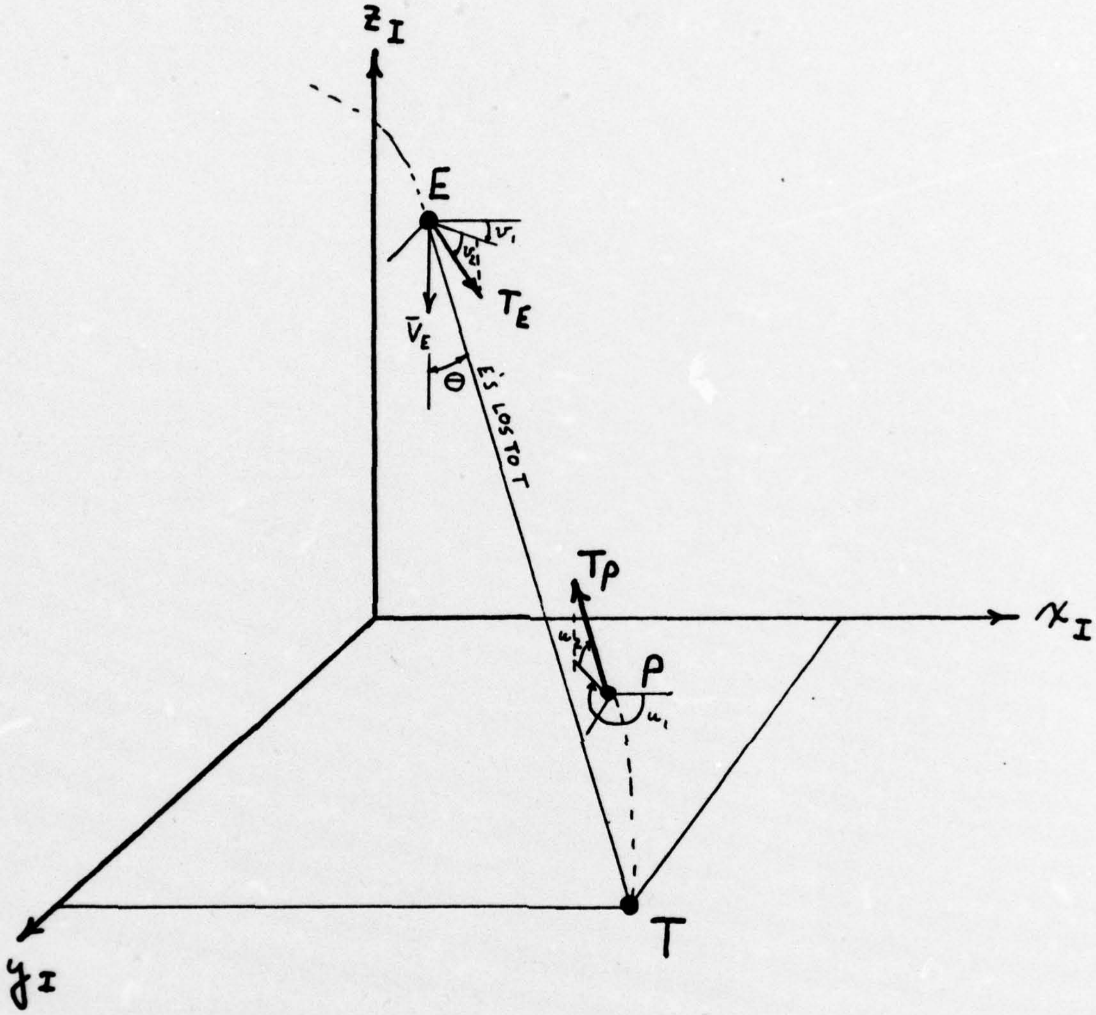


Figure 4. Interceptor-Penetrator Game Geometry

<u>Equations of Motion</u>. The state equations and initial conditions are

$$
\begin{aligned}
\dot{X} &= V_x\,, & X(0) &= 0. \\
\dot{V}_x &= T_E \cos v_2 \cos v_1\,, & V_x(0) &= .57735 \\
\dot{Y} &= V_y\,, & Y(0) &= 0. \\
\dot{V}_y &= T_E \cos v_2 \sin v_1\,, & V_y(0) &= .57735 \\
\dot{Z} &= V_z\,, & Z(0) &= 20. \\
\dot{V}_z &= T_E \sin v_2\,, & V_z(0) &= -.57735 \\
\dot{x} &= v_x\,, & x(0) &= 20. \\
\dot{v}_x &= T_p \cos u_2 \cos u_1\,, & v_x(0) &= 0. \\
\dot{y} &= v_y\,, & y(0) &= 20. \\
\dot{v}_y &= T_p \cos u_2 \sin u_1\,, & v_y(0) &= 0. \\
\dot{z} &= v_z\,, & z(0) &= 0. \\
\dot{v}_z &= T_p \sin u_2\,, & v_z(0) &= 0.
\end{aligned}
\qquad (36)
$$

The target coordinates are $(X_T\ , Y_T\ , Z_T) = (20\ , 20\ , 0)$ d.u.

Note that the initial conditions are such that E's velocity vector at t = 0 is directed at T with a magnitude of 1.0 d.u./t.u. (this is equivalent to 10,000 feet/second). At t = 0, P is positioned at T with zero initial velocity.

In this problem it is assumed that the two vehicles are point masses operating in a drag free environment with constant thrust to mass ratios. Further, it is assumed that the acceleration of gravity is negligible in comparison

to the acceleration capabilities of the vehicles.

Cost. The cost used in this game consists of one half the square of the final EP range plus the dot product of E's velocity vector and E's line of sight distance vector to T at the final time. The angle Θ between these two vectors is indicative of the predicted ET miss distance should E successfully survive the encounter with P. A small value of Θ at the final time would indicate that E is in a favorable position from which to later impact T. Conversely, a large value of $\Theta(t_f)$ would indicate that E is in a poor position from which to hit T. Further discussion of this aspect of the problem may be found in Appendix B.

The cost is given by

$$J = \tfrac{1}{2}\left[(X-x)^2 + (Y-y)^2 + (Z-z)^2 \right]\Big|_{t_f} + \left[V_X (X_T - X) + V_Y (Y_T - Y) + V_Z (Z_T - Z) \right]\Big|_{t_f} \quad . \qquad (37)$$

E is the maximizing player and P is the minimizing player.

Application of Necessary Conditions

The Hamiltonian for this problem is

$$H = \lambda_X V_X + \lambda_{V_X} T_E \cos v_2 \cos v_1 + \lambda_Y V_Y + \lambda_{V_Y} T_E \cos v_2 \sin v_1 + \lambda_Z V_Z + \lambda_{V_Z} T_E \sin v_2 + \lambda_x v_x + \lambda_{v_x} T_P \cos u_2 \cos u_1 + \lambda_y v_y + \lambda_{v_y} T_P \cos u_2 \sin u_1 + \lambda_z v_z + \lambda_{v_z} T_P \sin u_2 \quad . \qquad (38)$$

The adjoint differential equations and boundary conditions are found from Equations (6) and are

$$
\begin{aligned}
\dot{\lambda}_X &= 0 \quad &, \quad \lambda_X(t_f) &= (X - x - V_X)\Big|_{t_f} \\
\dot{\lambda}_{V_X} &= -\lambda_X \quad &, \quad \lambda_{V_X}(t_f) &= X_T - X_f \\
\dot{\lambda}_Y &= 0 \quad &, \quad \lambda_Y(t_f) &= (Y - y - V_Y)\Big|_{t_f} \\
\dot{\lambda}_{V_Y} &= -\lambda_Y \quad &, \quad \lambda_{V_Y}(t_f) &= Y_T - Y_f \\
\dot{\lambda}_Z &= 0 \quad &, \quad \lambda_Z(t_f) &= (Z - z - V_Z)\Big|_{t_f} \\
\dot{\lambda}_{V_Z} &= -\lambda_Z \quad &, \quad \lambda_{V_Z}(t_f) &= Z_T - Z_f \\
\dot{\lambda}_x &= 0 \quad &, \quad \lambda_x(t_f) &= x_f - X_f \\
\dot{\lambda}_{v_x} &= -\lambda_x \quad &, \quad \lambda_{v_x}(t_f) &= 0 \\
\dot{\lambda}_y &= 0 \quad &, \quad \lambda_y(t_f) &= y_f - Y_f \\
\dot{\lambda}_{v_y} &= -\lambda_y \quad &, \quad \lambda_{v_y}(t_f) &= 0 \\
\dot{\lambda}_z &= 0 \quad &, \quad \lambda_z(t_f) &= z_f - Z_f \\
\dot{\lambda}_{v_z} &= -\lambda_z \quad &, \quad \lambda_{v_z}(t_f) &= 0 \quad . \qquad (39)
\end{aligned}
$$

From the optimal control conditions, Equations (7) through (10), it is found that

$$
\begin{aligned}
\sin v_1 &= \frac{\lambda_{v_Y}}{\sqrt{\lambda_{v_X}^2 + \lambda_{v_Y}^2}} \quad &, \quad \sin w_1 &= \frac{-\lambda_{v_y}}{\sqrt{\lambda_{v_x}^2 + \lambda_{v_y}^2}} \\
\cos v_1 &= \frac{\lambda_{v_X}}{\sqrt{\lambda_{v_X}^2 + \lambda_{v_Y}^2}} \quad &, \quad \cos w_1 &= \frac{-\lambda_{v_x}}{\sqrt{\lambda_{v_x}^2 + \lambda_{v_y}^2}} \\
\sin v_2 &= \frac{\lambda_{v_Z}}{\lambda_{v_X}^2 + \lambda_{v_Y}^2 + \lambda_{v_Z}^2} \quad &, \quad \sin w_2 &= \frac{-\lambda_{v_z}}{\sqrt{\lambda_{v_x}^2 + \lambda_{v_y}^2 + \lambda_{v_z}^2}} \\
\cos v_2 &= \frac{\sqrt{\lambda_{v_X}^2 + \lambda_{v_Y}^2}}{\sqrt{\lambda_{v_X}^2 + \lambda_{v_Y}^2 + \lambda_{v_Z}^2}} \quad &, \quad \cos w_2 &= \frac{\sqrt{\lambda_{v_x}^2 + \lambda_{v_y}^2}}{\sqrt{\lambda_{v_x}^2 + \lambda_{v_y}^2 + \lambda_{v_z}^2}} . \qquad (40)
\end{aligned}
$$

In this problem, the final time is established when

$$H(t_f) = 0 \quad . \tag{41}$$

Equation (41) is equivalent to

$$\dot{J} = 0 \quad . \tag{42}$$

With the initial conditions given in Equations (36) and the payoff given by Equation (37), it was found that the optimal open-loop strategies called for E to thrust directly towards T and for P to thrust directly towards E. This strategy is obviously suicidal for E, hence, if E is to have any hope of destroying T, he must first avoid P.

Suppose, in an effort to evade P, E uses the non-optimal control $v_{NONOPT} = (0,0)$ for the duration of the encounter with P. This control is unknown to P. In order to account for E's non-optimal play, P uses a closed-loop control law discussed in Chapter II.

Closed-Loop Controls

The first algorithm used was the gradient scheme with control updates given by Equations (19) and (20). This method was implemented on a real-time basis. The second algorithm used was a combined gradient-DDP scheme. The gradient-DDP method was implemented in a closed-loop but

not real-time fashion.

The convergence criterion used in all runs was

$$\|H_u(t_s)\|^2 \leq \epsilon \qquad \text{and}$$

$$\|H_v(t_s)\|^2 \leq \epsilon \quad . \qquad (43)$$

If this convergence criterion was not met then computation was terminated after $KOUNT_{MAX}$ iterations. The game was played using various iteration cutoff factors ($KOUNT_{MAX}$) and sampling intervals (Δ).

Thrust to mass ratios were also varied in this game. Values of T_P range from about 93 g's (0.3 d.u./t.u.^{2}) to 310 g's (1.0 d.u./t.u.^{2}). These are not unreasonable figures for an ABM missile such as the Sprint (Ref 6:5).

The initial nominal control guesses used were $u_o(0)=(-135°,35.26°)$ and $v_o(0)=(45°,-35.26°)$. These values were used by the algorithm on the first iteration at the first sampling time.

Results

Gradient Method. The gradient closed-loop control law made use of Equations (19) and (20) in computing the control corrections δu and δv. This method was implemented in real-time. In all runs, player E used the non-optimal control

$v_{NONOPT} = (0,0)$.

The changes in the cost and minimum EP range were studied for various sampling intervals (Δ) and iteration cutoff factors ($KOUNT_{MAX}$). In all of these runs the thrust to mass ratios were fixed at $T_P = 1.0$; $T_E = 0.1$ and the convergence factor was set at $\epsilon = 0.5$. For purposes of discussion, a minimum EP range of 1.0 d.u. or less is scored as a "kill" for P. The results are presented in Figures 5 through 8.

Note from these figures that the minimum EP range increases substantially if too small a sampling interval is used. This occurs because with very small sampling intervals there may not be sufficient time between sampling times for P to implement the updated control. This effect is especially apparent early in the game because the algorithm usually required more iterations (and more time) to meet the convergence criterion. For example, in a run with $\Delta = 0.4$ and $KOUNT_{MAX} = 4$, player P was only able to implement the updated controls in the last 4.3 t.u.'s (seconds) of the game. This represents only about 35% of the total flight time.

If Δ was made too large the cost and minimum EP range increased. This is caused by P not updating his control frequently enough.

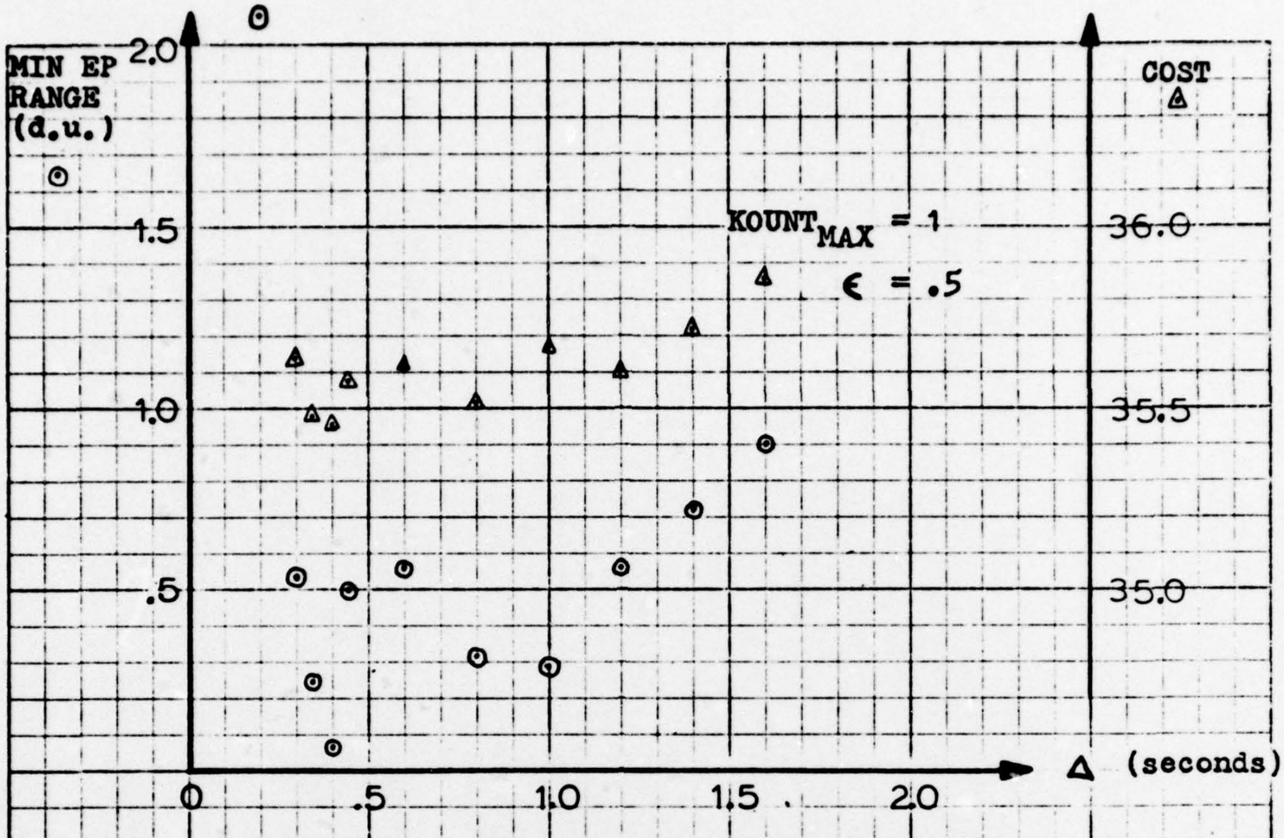


Figure 5

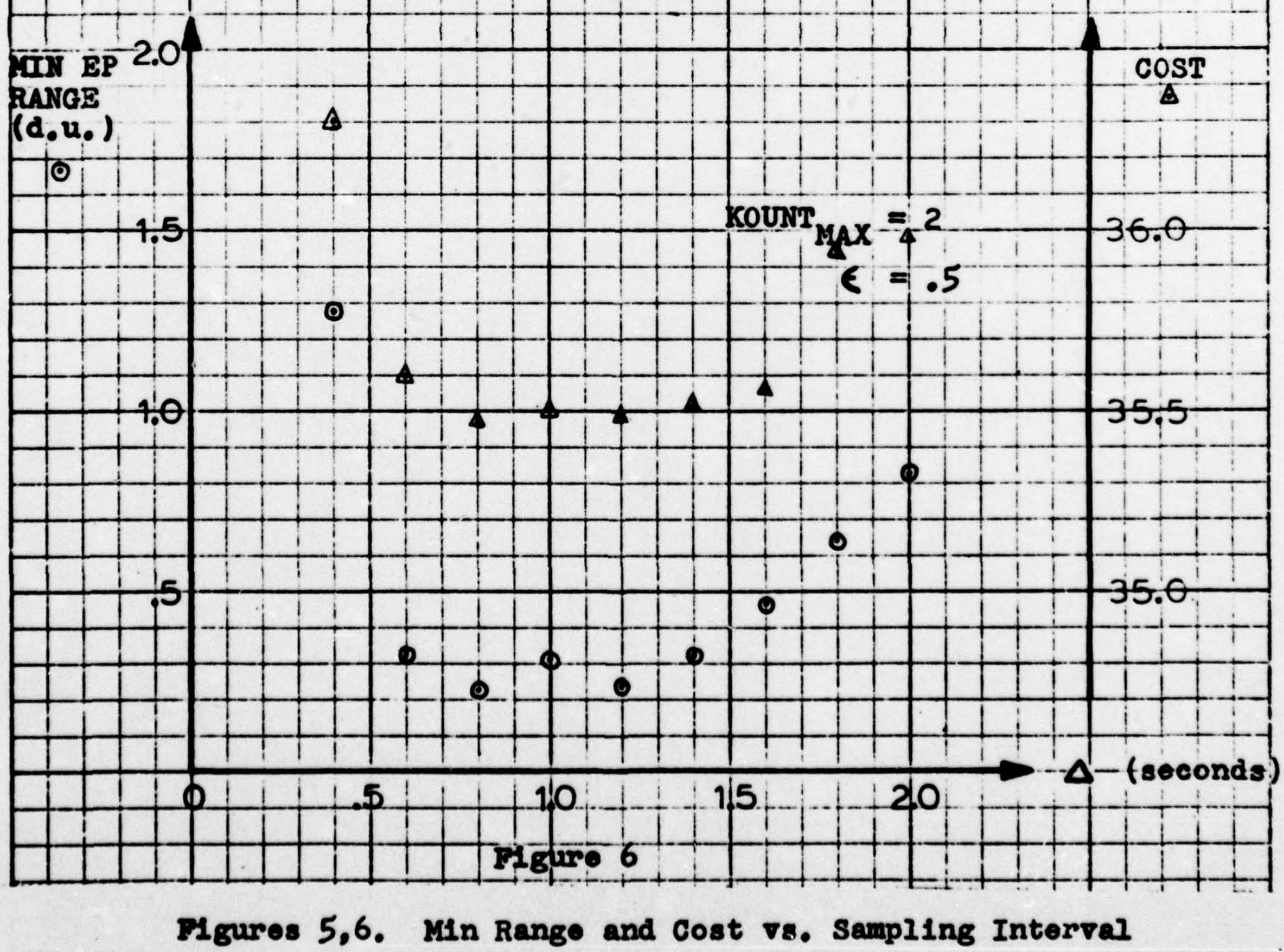


Figure 6

Figures 5,6. Min Range and Cost vs. Sampling Interval

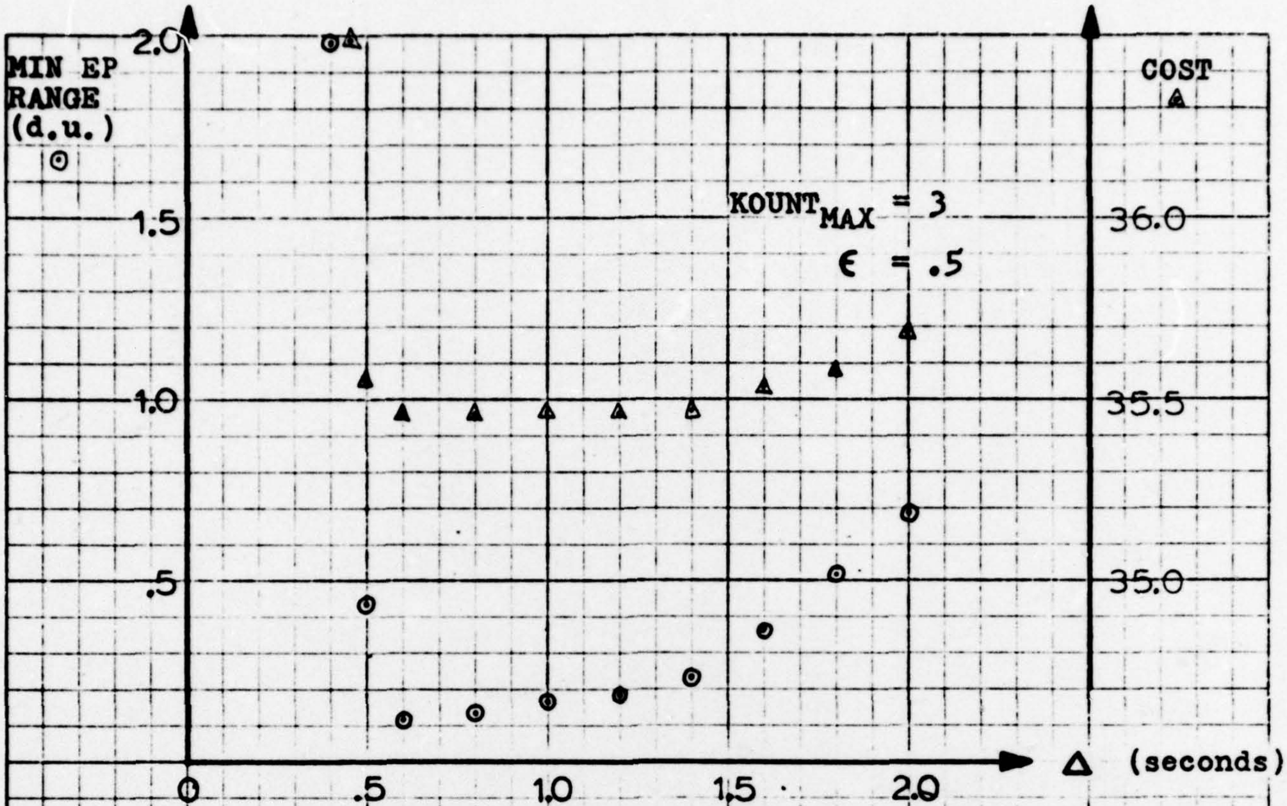


Figure 7

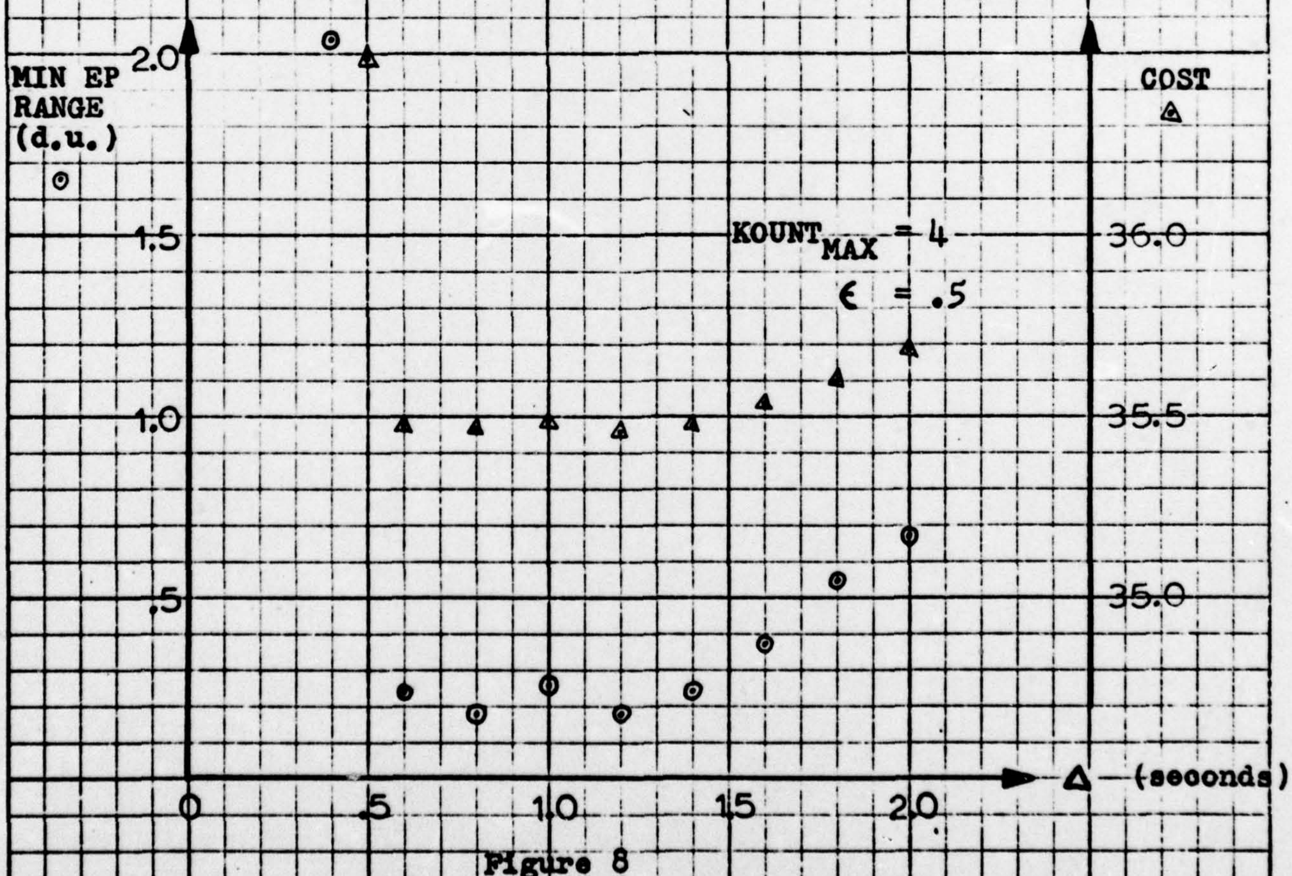


Figure 8

Figures 7,8 . Min Range and Cost vs. Sampling Interval

It was found that the cost and minimum EP range curves dropped to a minimum for a small range of sampling intervals. This range extended from about $\Delta = 0.5$ to $\Delta = 1.5$. The best overall results, from P's standpoint, were obtained for a sampling interval range of about 0.5 to 1.5 and iteration cutoff factors ($KOUNT_{MAX}$) of 3 or 4.

The run with $KOUNT_{MAX} = 1$ and $\Delta = 0.4$ (Figure 5) gave the lowest minimum EP range of any run, however, this was an isolated case. Other sampling intervals with $KOUNT_{MAX} = 1$ resulted in erratic behavior of the cost and minimum EP range.

It was found on all runs that only one or two iterations of the algorithm were required at each sampling time beyond about $t_f/2$. This suggests that a smaller Δ could have been used as the game progressed.

In order to investigate the sensitivity of the algorithm to the initial nominal control guess and initial conditions, the initial nominal control values and initial conditions of E were fixed and the position of the target (and initial position of P) were changed. The results are summarized in Table 3. Note that P was able to compensate and score a "kill" for cases 1 and 2 but not for case 3. In case 2, the fact that the algorithm was able to recover from the relatively poor initial control guesses and allow P to score a "kill" shows some degree of insensitivity of the method to poor initial control guesses.

Table 3. Sensitivity to Initial Conditions and Initial Control Guesses.			
Case	Target Coordinates	Min EP Range	t_f
1	$X_T = 20$ $Y_T = 20$ $Z_T = 0$	.43160	11.755
2	$X_T = 25$ $Y_T = 25$ $Z_T = 0$	.96026	13.083
3	$X_T = 30$ $Y_T = 30$ $Z_T = 0$	3.22657	14.362

Factors Common to all Three Cases:

$T_p = .3$ (93g's) $\quad \Delta = .8 \quad v_{NONOPT} = (0,0)$

$T_e = .1$ (31g's) $\quad \epsilon = .5 \quad KOUNT_{MAX} = 3$

Initial Control Guesses $\quad u_o(0) = (-135^\circ, 35.26^\circ)$

$v_o(0) = (45^\circ, -35.26^\circ)$

E's Initial Conditions $\quad X(0) = 0. \quad Y(0) = 0. \quad Z(0)=20$

$V_X(0)=.57735 \quad V_Y(0)=.57735 \quad V_Z(0)=-.5773$

P's Initial Conditions $\quad$ P located at target with zero initial velocity

Runs of the game were made with various values of T_e and T_p. The results are summarized in Table 4 below. Note that P was able to score a kill for a fairly wide range of values of T_e (from -62.1 g's to about +55.9 g's).

Table 4. Sensitivity to Thrust Levels

T_p	T_e	Min EP Range (d.u.)	t_f (sec)
1.00 (310.5g's)	-1.00 (-310.5g's)	25.656*	6.000
"	- .20 (-62.1 g's)	.292	8.056
"	- .10 (-31.1 g's)	.424	7.636
"	0.00 (0.0 g's)	.236	7.374
"	.10 (31.1 g's)	.173	7.287
"	.12 (37.3 g's)	.377	7.197
"	.14 (43.5 g's)	.510	7.159
"	.16 (49.7 g's)	.652	7.115
"	.18 (55.9 g's)	.911	7.124
"	.20 (62.1 g's)	1.241*	7.084
"	.40 (124.2 g's)	4.306*	6.800
"	.60 (186.3 g's)	7.304*	6.489

Factors Common to all of the Above Cases

$\Delta = 1.2$ $\quad v_{NONOPT} = (0,0)$

$\epsilon = .5$ $\quad KOUNT_{MAX} = 4$ $\quad (X_T, Y_T, Z_T) = (20, 20, 0)$

*E has escaped P.

Combined Gradient-DDP Method (Not Real-Time Implementation). In an effort to achieve a more accurate solution (and a lower minimum EP range) to the interceptor-penetrator problem, a guidance scheme using both the gradient and DDP methods at each t_s was devised. The control update algorithm made use of a maximum of 10 iterations of the gradient scheme and then a maximum of 10 iterations of the DDP scheme. It was found that the DDP scheme made extremely large control changes early in the encounter. Also during that time, the predicted cost change was on the order of 10^2 rather than zero. Very late in the game the method "settled down" but at the expense of a large predicted t_f which resulted from the poor control updates made earlier. A run using $\Delta = 1.0$ t.u. resulted in a minimum EP range of 0.455 d.u. and a t_f of 12.39 t.u. This is no improvement over using the gradient routine alone.

In the above mechanization, it is apparent that early in the game the nominal controls "handed over" to the DDP scheme by the gradient scheme were not sufficiently accurate for convergence of the DDP. This is due in part to the lack of any convergence control built into the DDP scheme and in part, due to the relatively high thrust rockets being used in the game. Because of the high thrusts, a very small error in a control angle results

in a large cost change at the predicted t_f.

Another combined gradient-DDP method was tried in which the gradient routine alone was used early in the game ($0 \leq t \leq .9t_f$) with a $\Delta = 1.0$ t.u. and then the DDP routine alone was used late in the game ($.9t_f < t \leq t_f$) with a $\Delta = .082$ t.u. This resulted in $t_f = 10.92$ t.u. and a minimum EP range of 3.28 d.u. Thus, P did not score a kill in this case. Again, it was found that DDP did not converge. It is apparent that if DDP is to be used in conjunction with the gradient scheme then DDP needs some built-in convergence control device. The nominal controls handed over to DDP by the gradient scheme were fairly close to the optimal and yet with no convergence control the DDP scheme failed. Use of the gradient method alone provided better results because of its inherent stability.

V. Conclusions

It has been shown that a closed-loop control algorithm based on a gradient method can provide reasonably fast and accurate solutions for some differential game problems. The principle advantages of this method are that it is stable, it depends only on the sampled system state and not a reference trajectory, and it provides reasonable accuracy within a few iterations. The main disadvantages are that the method does require some degree of fine tuning in selecting the step-size matrices K_P , K_E and the convergence factor, ϵ . Also, the gradient method does not provide a highly accurate solution. However, in some problems, speed of computation is more important than high accuracy. A DDP method used with the gradient method may provide higher accuracy but some device needs to be added to limit the size of the control corrections made on iteration. The DDP method as outlined in Chapter II was found to be highly unstable. The gradient closed-loop control law presented in this thesis provides an alternative to methods that may be prove to stability or convergence problems.

Recommendations. Use of a closed-loop real-time gradient method on a problem with realistic dynamics should be tried and the results compared to those obtained using other guidance laws. Also, more work may be done on the

hybrid gradient-DDP scheme to achieve more accuracy of the solution and better convergence of the DDP. Another area of interest is the formulation of an interceptor-penetrator duel between an aircraft and a surface-to-air missile using realistic dynamics.

Bibliography

1. Anderson, G.M., "A Near-Optimal Closed-Loop Solution Method for Nonsingular Zero-Sum Differential Games." Journal of Optimization Theory and Applications, 13: 303-318 (March 1974).

2. Anderson, G.M. and G.D. Bohn, "A Near-Optimal Closed-Loop Control Law for Pursuit-Evasion Problems Between Two Spacecraft." AIAA Paper 76-794, AIAA-AAS Astrodynamics Conference, San Diego, California, August 18-20, 1976.

3. Bryson, A.E. and Y.C. Ho, Applied Optimal Control. New York: The Halstead Press, 1975.

4. Leatham, A.L. and U.H.D. Lynch, "Two Numerical Methods to Solve Realistic Air-to-Air Combat Differential Games." AIAA Paper 74-22, Washington D.C., 1974.

5. Jarmark, B.S.A., "Convergence Control in Differential Dynamic Programming Applied to Air-to-Air Combat." AIAA Journal, 14: 118-120 (January 1976).

6. Barnaby, C.F., "The Development and Characteristics of Anti-Ballistic Missile Systems." Implications of Anti-Ballistic Missile Systems, Pugwash Symposium, 1969, edited by C.F. Barnaby and A. Boserup, New York: The Humanities Press, 1969.

7. Powell, M.J.D., "A FORTRAN Subroutine for Solving Systems of Non-Linear Algebraic Equations." Numerical Methods for Non-Linear Algebraic Equations, edited by Philip Rabinowitz, New York: Gordon and Breach Science Publishers, 1970.

Appendix A

Derivation of an Analytic Closed-Loop Control Law

An analytic expression for the optimal closed-loop control law to be used by P in the problem of Chapter III may be found by noting that the adjoint variables (Equations (32)) are linear in the time-to-go ($\tau = t_f - t$). Also it is noted that the controls for E and P are equal as given by Equations (33). In order to find analytic expressions for the adjoint variables to use in (33), one needs to find X_f and Y_f. Eliminating the controls from the second state equation in (28) one has,

$$\dot{V}_x = (T_E - T_P) \frac{X_f}{\sqrt{X_f^2 + Y_f^2}} = K_1 \quad , \qquad \text{(A-1)}$$

where K_1 is a constant. Integration of (A-1) with respect to τ yields

$$V_{x_f} - V_x = K_1 (t_f - t) \quad . \qquad \text{(A-2)}$$

Rearranging gives

$$V_x = \dot{X} = -K_1 (t_f - t) + V_{xf} \quad ,$$

which may be integrated with respect to τ to give

$$X_f - X = -\frac{K_1}{2}(t_f - t)^2 + V_{Xf}(t_f - t) \quad . \quad \text{(A-3)}$$

By substituting for V_{Xf} from (A-2) into (A-3) one has

$$X_f = X + V_X(t_f - t) + \frac{K_1}{2}(t_f - t)^2 \quad . \quad \text{(A-4)}$$

Performing similar integrations for Y_f yields

$$Y_f = Y + V_Y(t_f - t) + \frac{K_2}{2}(t_f - t)^2 \quad , \quad \text{(A-5)}$$

where K_2 is a constant $K_2 = (T_E - T_P)\frac{Y_f}{\sqrt{X_f^2 + Y_f^2}}$.

Substituting for K_1 in (A-4) and multiplying by Y_f yields

$$X_f Y_f = [X + V_X(t_f - t)]\,Y_f \; + \frac{(t_f - t)^2}{2}(T_E - T_P)\frac{X_f Y_f}{\sqrt{X_f^2 + Y_f^2}} \quad . \quad \text{(A-6)}$$

Substituting for K_2 in (A-5) and multiplying by X_f gives

$$X_f Y_f = [Y + V_Y(t_f - t)]\,X_f \; + \frac{(t_f - t)^2}{2}(T_E - T_P)\frac{X_f Y_f}{\sqrt{X_f^2 + Y_f^2}} \quad . \quad \text{(A-7)}$$

Subtracting (A-7) from (A-6) and letting

$$\bar{X} = X + V_x(t_f - t) \quad ;$$

$$\bar{Y} = Y + V_y(t_f - t)$$

gives (A-8)

$$\bar{X} Y_f = \bar{Y} X_f \quad . \tag{A-9}$$

Equation (A-9) is used to obtain expressions for λ_{v_x} and λ_{v_y} in terms of the current state and time-to-go. Substitution of λ_{v_x} and λ_{v_y} into Equations (33) then gives

$$\cos u^* = \frac{\bar{X}}{\sqrt{\bar{X}^2 + \bar{Y}^2}} \quad ;$$

$$\sin u^* = \frac{\bar{Y}}{\sqrt{\bar{X}^2 + \bar{Y}^2}} \quad , \tag{A-10}$$

which is an analytic closed-loop control law.

Appendix B

ABM-ICBM Duel

Minimization of the ET Miss Distance. If the ICBM (designated with an E) successfully survives the encounter with the ABM (designated with a P) then E will thrust so as to minimize the ET distance. Assuming that E can thrust continuously, the system dynamics are

$$
\begin{aligned}
\dot{X} &= V_X \\
\dot{V}_X &= T_E \cos v_2 \cos v_1 \\
\dot{Y} &= V_Y \\
\dot{V}_Y &= T_E \cos v_2 \sin v_1 \\
\dot{Z} &= V_Z \\
\dot{V}_Z &= T_E \sin v_2
\end{aligned}
\quad . \qquad \text{(B-1)}
$$

The initial conditions for this problem are determined by the final state of E at the termination of the EP encounter. The cost is

$$\underset{v}{MIN}\ J = \frac{1}{2}\left[(X - X_T)^2 + (Y - Y_T)^2 + (Z - Z_T)^2\right]\Big|_{T_f} \quad . \qquad \text{(B-2)}$$

where T_f is established when

$$\dot{J} \rightarrow 0 \quad . \qquad \text{(B-3)}$$

There are no terminal state constraints in this problem formulation. The controls, v , are the same as those

described in Chapter IV.

The Hamiltonian for this problem is

$$H = \lambda_x V_x + \lambda_{Vx} T_E \cos v_2 \cos v_1 + \lambda_Y V_Y + \lambda_{VY} T_E \cos v_2 \sin v_1 +$$

$$\lambda_Z V_Z + \lambda_{VZ} T_E \sin v_2 \quad . \quad \text{(B-4)}$$

The costate differential equations and terminal boundary conditions are

$$\begin{aligned}
\dot{\lambda}_x &= 0 , & \lambda_x(T_f) &= X(T_f) - X_T \\
\dot{\lambda}_{Vx} &= -\lambda_X , & \lambda_{Vx}(T_f) &= 0 \\
\dot{\lambda}_Y &= 0 , & \lambda_Y(T_f) &= Y(T_f) - Y_T \\
\dot{\lambda}_{VY} &= -\lambda_Y , & \lambda_{VY}(T_f) &= 0 \\
\dot{\lambda}_Z &= 0 , & \lambda_Z(T_f) &= Z(T_f) - Z_T \\
\dot{\lambda}_{VZ} &= -\lambda_Z , & \lambda_{VZ}(T_f) &= 0
\end{aligned} \quad . \quad \text{(B-5)}$$

The value of the Hamiltonian at the final time is

$$H(T_f) = 0 \quad . \quad \text{(B-6)}$$

Application of the optimality conditions $H_v = 0$ and $H_{vv} \geq 0$ yields

$$\sin v_1 = \frac{-\lambda_{VY}}{\sqrt{\lambda_{Vx}^2 + \lambda_{VY}^2}} , \quad \sin v_2 = \frac{-\lambda_{VZ}}{\sqrt{\lambda_{Vx}^2 + \lambda_{VY}^2 + \lambda_{VZ}^2}}$$

$$\cos v_1 = \frac{-\lambda_{Vx}}{\sqrt{\lambda_{Vx}^2 + \lambda_{VY}^2}} , \quad \cos v_2 = \frac{\sqrt{\lambda_{Vx}^2 + \lambda_{VY}^2}}{\sqrt{\lambda_{Vx}^2 + \lambda_{VY}^2 + \lambda_{VZ}^2}} \quad . \quad \text{(B-7)}$$

Expressions for the costates in (B-7) are found from (B-5) and are

$$\lambda_{v_x} = -\left(X(T_f) - X_T\right)\left(T_f - t\right)$$

$$\lambda_{v_y} = -\left(Y(T_f) - Y_T\right)\left(T_f - t\right)$$

$$\lambda_{v_z} = -\left(Z(T_f) - Z_T\right)\left(T_f - t\right) \quad . \quad \text{(B-8)}$$

Equations for $X(T_f) - X_T$, $Y(T_f) - Y_T$, $Z(T_f) - Z_T$ are found by integrating the second, fourth, and sixth state equations with respect to the time-to-go ($T_f - t$). Substituting these into (B-8) then using Equations (B-8) in (B-7) after some manipulation yields

$$\sin v_1 = \frac{-\bar{Y}}{\sqrt{\bar{X}^2 + \bar{Y}^2}}$$

$$\cos v_1 = \frac{-\bar{X}}{\sqrt{\bar{X}^2 + \bar{Y}^2}}$$

$$\sin v_2 = \frac{-\bar{Z}}{\sqrt{\bar{X}^2 + \bar{Y}^2 + \bar{Z}^2}}$$

$$\cos v_2 = \frac{\sqrt{\bar{X}^2 + \bar{Y}^2}}{\sqrt{\bar{X}^2 + \bar{Y}^2 + \bar{Z}^2}} \quad , \quad \text{(B-9)}$$

where

$$\bar{X} = X - X_T + V_x\left(T_f - t\right)$$

$$\bar{Y} = Y - Y_T + V_y\left(T_f - t\right)$$

$$\bar{Z} = Z - Z_T + V_z\left(T_f - t\right) \quad . \quad \text{(B-10)}$$

In order to use (B-9) and (B-10), a value of T_f must be established. If it is assumed that T_f occurs approximately when E impacts the earth then a series of fixed time problems may be solved where a guessed T_f is incremented until $Z(T_f) \leq 0$.

Interceptor-Penetrator Problem Program Listing. The following program listing was used to investigate the minimization of the ET impact distance should E sucessfully avoid P. This same program was used to generate open-loop strategies for this game using NS01A (Ref 7):

```
RRB,T40,STCS P.        T760203,BACON,BOX 4087
FTN.
ATTACH,A,AFITSUBROUTINES,ID=AFIT,SN=AFIT.
ATTACH,B,ASPOLIB,ID=T760004.
LIBRARY,B,A.
LGO.
      PROGRAM GAME2(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
      DIMENSION X(12),F(12),AJINV(12,12),W(600)
      COMMON /BLOK/ TE,TP,XT,YT,ZT,TF,SINU1,COSU1,SINU2,COSU2,
     CSINV1,COSV1,SINV2,COSV2
C     THIS RUN MADE WITH THE EVADER PLAYING NONOPTIMALLY
C     USING CONTROLS V=(0,0)
      X(1) = -6.
      X(2) = -50.
      X(3) = -6.
      X(4) = -50.
      X(5) = 6.
      X(6) = 50.
      X(7) = 6.
      X(8) = 50.
      X(9) = 6.
      X(10) = 50.
      X(11) = -6.
      X(12) = -60.
      TF = 11.53
      N=12
      DSTEP = 1.E-6
      DMAX=1.E9
      ACC=1.E-6
      MAXFUN = 150
      IPRINT = 1
      CALL NS01A(N,X,F,AJINV,DSTEP,DMAX,ACC,MAXFUN,IPRINT,W)
      FCK = 0.
      DO 10 I=1,12
10    FCK = FCK + F(I)**2
      IF(FCK.GT.ACC) STOP
      CALL OPTTRAJ(X)
      END
      SUBROUTINE CALFUN(N,X,F)
      DIMENSION X(N),F(N),Y(24)
      COMMON /BLOK/ TE,TP,XT,YT,ZT,TF,SINU1,COSU1,SINU2,COSU2,
     CSINV1,COSV1,SINV2,COSV2
      EXTERNAL SLOPE
      DO 5 I=1,12
5     Y(I) = 0.
      Y(2) = .57735
      Y(4) = .57735
      Y(5) = 20.
      Y(6) =-.57735
      Y(7) = 20.
      Y(9) = 20.
      TE = .1
      TP = .3
      XT = 20.
      YT = 20.
      ZT = 0.
      II=12
```

```
      DO 10 I=1,12
      II=II+1
10    Y(II)=X(I)
      T=0.
      M=24
      DT = TF/128.
      CALL SET(M,T,Y,DT,SLOPE,D,.T.,D,D)
      DO 20 J=1,128
      CALL STEP(M,T,Y,DT,SLOPE,D,.T.,D,D)
20    CONTINUE
      F(1) = Y(13)+Y(2)-(Y(1)-Y(7))
      F(2) = Y(14)+(Y(1)-XT)
      F(3) = Y(15)+Y(4)-(Y(3)-Y(9))
      F(4) = Y(16)+(Y(3)-YT)
      F(5) = Y(17)+Y(6)-(Y(5)-Y(11))
      F(6) = Y(18)+(Y(5)-ZT)
      F(7) = Y(19)-(Y(7)-Y(1))
      F(8) = Y(20)
      F(9) = Y(21)-(Y(9)-Y(3))
      F(10) = Y(22)
      F(11) = Y(23)-(Y(11)-Y(5))
      F(12) = Y(24)
      END
      SUBROUTINE SLOPE(M,T,Y,DY)
      DIMENSION Y(M),DY(M)
      COMMON/BLOK/ TE,TP,XT,YT,ZT,TF,SINU1,COSU1,SINU2,COSU2,
     CSINV1,COSV1,SINV2,COSV2
C     E PLAYS NON-OPTIMALLY WITH CONTROLS V=(0,0).
      SINV1 = 0.
      COSV1 = 1.
      SINV2 = 0.
      COSV2 = 1.
      SQRTP1 = -SQRT(Y(20)**2+Y(22)**2)
      SQRTP2 = -SQRT(SQRTP1**2+Y(24)**2)
      SINU1 = Y(22)/SQRTP1
      COSU1 = Y(20)/SQRTP1
      SINU2 = Y(24)/SQRTP2
      COSU2 = SQRTP1/SQRTP2
      DY(1) = Y(2)
      DY(2) = TE*COSV2*COSV1
      DY(3) = Y(4)
      DY(4) = TE*COSV2*SINV1
      DY(5) = Y(6)
      DY(6) = TE*SINV2
      DY(7) = Y(8)
      DY(8) = TP*COSU2*COSU1
      DY(9) = Y(10)
      DY(10) = TP*COSU2*SINU1
      DY(11) = Y(12)
      DY(12) = TP*SINU2
      DY(13) = 0.
      DY(14) = -Y(13)
      DY(15) = 0.
      DY(16) =-Y(15)
      DY(17) = 0.
      DY(18) = -Y(17)
      DY(19) = 0.
      DY(20) = -Y(19)
```

```
      DY(21)  =  0.
      DY(22)  =  -Y(21)
      DY(23)  =  0.
      DY(24)  =  -Y(23)
      END
      SUBROUTINE OPTTRAJ(X)
      DIMENSION Y(24),X(12)
      COMMON /BLOK/ TE,TP,XT,YT,ZT,TF,SINU1,COSU1,SINU2,COSU2,
     CSINV1,COSV1,SINV2,COSV2
      EXTERNAL SLOPE
      DO 5 I =1,12
5     Y(I) = 0.
      Y(2) = .57735
      Y(4) = .57735
      Y(5) = 20.
      Y(6) = -.57735
      Y(7) = 20.
      Y(9) = Y(7)
      II = 12
      DO 10 I=1,12
      II = II+1
10    Y(II) = X(I)
      DT = TF/64.
      T = 0.
      M = 2+
      PI = ACOS(-1.)
      PRINT  100,T,U1DEG,U2DEG,V1DEG,V2DEG
      PRINT  200,(Y(I),I=1,12)
      CALL SET(M,T,Y,DT,SLOPE,D,.T.,D,D)
      DO 20 J=1,64
      CALL STEP(M,T,Y,DT,SLOPE,D,.T.,D,D)
      U1 = ATAN2(SINU1,COSU1)
      U2 = ATAN2(SINU2,COSU2)
      V1 = ATAN2(SINV1,COSV1)
      V2 = ATAN2(SINV2,COSV2)
      U1DEG  = U1*180./PI
      U2DEG  = U2*180./PI
      V1DEG  = V1*180./PI
      V2DEG  = V2*180./PI
      PRINT  100,T,U1DEG,U2DEG,V1DEG,V2DEG
      PRINT  200,(Y(I),I=1,12)
      EPRANGE=SQRT((Y(1)-Y(7))**2+(Y(3)-Y(9))**2+(Y(5)-Y(11))**2)           00
      XJ=Y(2)*(XT-Y(1))+Y(4)*(YT-Y(3))+Y(6)*(ZT-Y(5))+.5*EPRANGE**2         00
      ETRANGE=SQRT((Y(1)-XT)**2+(Y(3)-YT)**2+(Y(5)-ZT)**2)                  00
      XL = ETRANGE
      VE = SQRT(Y(2)**2+Y(4)**2+Y(6)**2)
      VEDOTXL = Y(2)*(XT-Y(1))+Y(4)*(YT-Y(3))+Y(6)*(ZT-Y(5))
      THFACT  = VEDOTXL/(VE*XL)
      IF(ABS(THFACT).GT.1.)  THFACT  =  1.
      XTHETA  =(ACOS(THFACT))*180./PI
      PRINT  300,XJ,EPRANGE,ETRANGE,XTHETA
100   FORMAT ("TIME=",F10.5,2X,"U=",2(F10.5,1X),2X,"V=",2(F10.5,1X))
200   FORMAT ("STATE=",6(F9.5,1X)/6X,6(F9.5,1X))
300   FORMAT ("COST=",F10.5,2X,"EPRANGE=",F10.5,2X,"ETRANGE=",F10.5,
     C2X,"THETA=",F10.5,"DEGREES",/)
20    CONTINUE
      CALL ETMISS(Y)
      END
```

```
      SUBROUTINE ETMISS(X)
      DIMENSION Y(24),X(24)
      COMMON/BLOK/ TE,TP,XT,YT,ZT,TF,SINU1,COSU1,SINU2,COSU2,
     CSINV1,COSV1,SINV2,COSV2
      COMMON/CRUD/ BTF
      EXTERNAL ETEQU
      PRINT*,"THE FOLLOWING IS THE PREDICTED TRAJECTORY THAT
     CE WOULD FOLLOW TO T IF E SUCCESSFULLY AVOIDED P."
      PI = ACOS(-1.)
      KKK = 0
      BTF = 3.+TF
1     DO 2 I=1,6
2     Y(I) = X(I)
      M=6
      DT = BTF/64.
      T = TF
      CALL SET(M,T,Y,DT,ETEQU,0,.T.,0,0)
      DO 10 J=1,64
      CALL STEP(M,T,Y,DT,ETEQU,0,.T.,0,0)
      V1 = ATAN2(SINV1,COSV1)
      V2 = ATAN2(SINV2,COSV2)
      V1DEG = V1*180./PI
      V2DEG = V2*180./PI
      IF(KKK.EQ.1) PRINT 100,T,(Y(L),L=1,6)
100   FORMAT("TIME=",F9.5 ,1X,"(X,VX,Y,VY,Z,VZ)=",6(F9.5 ,1X))
      IF(KKK.EQ.1) PRINT 200,V1DEG,V2DEG
200   FORMAT("V1=",G15.8,"DEGREES",2X,"V2=",G15.8,"DEGREES"/)
      IF(Y(5).LE.0.0.AND.KKK.EQ.1) STOP
10    CONTINUE
      IF(Y(5).LE.0.0) KKK=KKK+1
      IF(KKK.GT.1) GO TO 50
      BTF = BTF+.2
      GO TO 1
50    RETURN
      END
      SUBROUTINE ETEQU(M,T,Y,DY)
      DIMENSION Y(M),DY(M)
      COMMON/BLOK/ TE,TP,XT,YT,ZT,TF,SINU1,COSU1,SINU2,COSU2,
     CSINV1,COSV1,SINV2,COSV2
      COMMON/CRUD/ BTF
      XBAR = Y(1)-XT+Y(2)*(BTF-T)
      YBAR = Y(3)-YT+Y(4)*(BTF-T)
      ZBAR = Y(5)-ZT+Y(6)*(BTF-T)
      SQRRT1 = SQRT(XBAR**2+YBAR**2)
      SQRRT2 = SQRT(SQRRT1**2+ZBAR**2)
      COSV1 =-XBAR/SQRRT1
      SINV1 =-YBAR/SQRRT1
      COSV2 = SQRRT1/SQRRT2
      SINV2 =-ZBAR/SQRRT2
      DY(1) = Y(2)
      DY(2) = TE*COSV2*COSV1
      DY(3) = Y(4)
      DY(4) = TE*COSV2*SINV1
      DY(5) = Y(6)
      DY(6) = TE*SINV2
      END
```

Appendix C

Program Listing: A Closed-Loop Gradient Control Update Algorithm (Real-Time Implementation)

```
BRR,T15,STCSP.            T760203,BACON,BOX 4087
FTN.
ATTACH,A,AFITSUBROUTINES,ID=AFIT,SN=AFIT.
LIBRARY,A.
LGO.
      PROGRAM GAME2(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
      DIMENSION Y(12),P(12)
      COMMON/BLOK/TP,TE,TF,TS,TSLAST,DEL,U(2),V(2),VNONOPT(2),
     CUT(150,2),VT(150,2),KK,JF,PI,XT,YT,ZT,USTAR(150,2),VSTAR(150,2)
      TP = .3
      TE = .1
      PI = ACOS(-1.)
C     SET STATE VALUES AT T=0.
      DO 5 I=1,12
5     Y(I) = 0.
      Y(2) = .57735
      Y(4) = .57735
      Y(5) = 20.
      Y(6) = -.57735
      Y(7) = 20.
      Y(9) = 20.
C     GUESS THE FINAL TIME.
      TF =10.51111
      DT = TF/128.
      XT = 20.
      YT = 20.
      ZT = 0.
C     DEFINE P'S SAMPLING INTERVAL, DEL, TO BE
      DEL = 1.
C     SET E'S NONOPTIMAL CONTROL SEQUENCE TO BE
      VNONOPT(1) = 0.
      VNONOPT(2) = 0.
C     THE INITIAL NOMINAL CONTROLS ARE
      U(1) = -135.*PI/180.
      U(2) = 35.26*PI/180.
      V(1) = 45.*PI/180.
      V(2) = -35.26*PI/180.
C     THE NOMINAL CONTROL HISTORIES FOR USE IN GRAD   THE 1ST TIME ARE
      DO 10 I=1,150
      UT(I,1)= U(1)
      UT(I,2) = U(2)
      VT(I,1) = V(1)
10    VT(I,2) = V(2)
      TS = DEL
      T = 0.
      M = 12
      U1DEG = U(1)*180./PI
      U2DEG = U(2)*180./PI
      V1DEG = V(1)*180./PI
      V2DEG = V(2)*180./PI
      PRINT 100,T,(Y(I),I=1,12)
      PRINT 200,U1DEG,U2DEG,V1DEG,V2DEG
C     INTEGRATE THE STATE EQUATIONS FORWARD IN TIME
      CALL F(T,Y,P)
20    CALL RKDES(T,Y,M,DT)
      U1DEG = U(1)*180./PI
      U2DEG = U(2)*180./PI
```

```
      V1DEG = V(1)*180./PI
      V2DEG = V(2)*180./PI
      PRINT 100,T,(Y(I),I=1,12)
      PRINT 200,U1DEG,U2DEG,V1DEG,V2DEG
100   FORMAT(/"TIME=",F10.5,2X,"E'S STATE=",6(F8.5,1X),
     C/,17X,"P'S STATE=",6(F8.5,1X))
200   FORMAT("U=",2(F10.5,2X),"V=",2(F10.5,2X))
      TSCKL=TS-.05
      TSCKH=TS+.05
      IF(T.GE.TSCKL.AND.T .LE.TSCKH) CALL SECOND(TSTART)
      IF(T.GE.TSCKL.AND.T .LE.TSCKH) CALL GRAD(Y)
      IF(T.GE.TSCKL.AND.T .LE.TSCKH) CALL SECOND(TFINISH)
      REALT = TFINISH-TSTART
      IF(T.GE.(TSLAST+REALT)) CALL CUPDATE
      IF(T.GE.TSCKL.AND.T .LE.TSCKH) PRINT*,"REAL TIME SPENT IN
     CGRAD=",REALT
      EPRANGE=SQRT((Y(1)-Y(7))**2+(Y(3)-Y(9))**2+(Y(5)-Y(11))**2)
      XJ=Y(2)*(XT-Y(1))+Y(4)*(YT-Y(3))+Y(6)*(ZT-Y(5))+.5*EPRANGE**2
      ETRANGE=SQRT((Y(1)-XT)**2+(Y(3)-YT)**2+(Y(5)-ZT)**2)
      PRINT 300,XJ,EPRANGE,ETRANGE
300   FORMAT("COST=",F10.5,2X,"EPRANGE=",F10.5,2X,"ETRANGE=",F10.5)
      IF(T.LE.TF) GO TO 20
      END
      SUBROUTINE GRAD(Y)                                                00
      DIMENSION Y(12),XS(12), X(26),HU(150,2),HV(150,2),OUT(150,2),
     COVT(150,2)
      COMMON/BLOK/TP,TE,TF,TS,TSLAST,DEL,U(2),V(2),VNONOPT(2),
     CUT(150,2),VT(150,2),KK,JF,PI,XT,YT,ZT,USTAR(150,2),VSTAR(150,2)
      EXTERNAL SLOPEF,SLOPEB                                            00
      KOUNT=0                                                           00
      XJDOTL = 0.0
      DU1MAX = 3.0*PI/180.
      DU2MAX = 3.0*PI/180.
      DV1MAX = 3.0*PI/180.
      DV2MAX = 3.0*PI/180.
      HUMAX1 = 1.
      HUMAX2 = 1.
      HVMAX1 = 1.
      HVMAX2 = 1.
5     KOUNT=KOUNT+1                                                     00
      DO 10 I=1,12                                                      00
10    XS(I)=Y(I)                                                        00
      T=TS                                                              00
      MF=12                                                             00
      DTF=(TF-TS)/64.                                                   00
      CALL SET(MF,T,XS,DTF,SLOPEF,0,.T.,0,0)                            00
      IF=0.                                                             00
20    IF=IF+1                                                           00
      JF=IF                                                             00
      CALL STEP(MF,T,XS,DTF,SLOPEF,0,.T.,0,0)                           00
      DDOT=(XS(1)-XS(7))*(XS(2)-XS(8))+(XS(3)-XS(9))*(XS(4)-XS(10))+    00
     C(XS(5)-XS(11))*(XS(6)-XS(12))                                     00
      FDOT =-XS(2)**2+(XT-XS(1))*TE*COS(VT(JF,2))*COS(VT(JF,1))
     C-XS(4)**2+(YT-XS(3))*TE*COS(VT(JF,2))*SIN(VT(JF,1))
     C-XS(6)**2+(ZT-XS(5))*TE*SIN(VT(JF,2))
      XJDOT = DDOT + FDOT
      IF(XJDOTL*XJDOT.LT.0.0) GO TO 25
      IF(JF.GE.150) GO TO 25
```

```
      XJDOTL = XJDOT
      GO TO 20                                                          00
25    TF = T
      PRINT 1000,XJDOT,JF,TF
1000  FORMAT ("XJDOT=",G15.8,"JF=",I8,"TF=",G15.8)
      DO 30 KT=1,12
30    X(KT) = XS(KT)
      X(13) = XS(1)-XS(7)-XS(2)
      X(14) = XT-XS(1)
      X(15) = XS(3)-XS(9)-XS(4)
      X(16) = YT-XS(3)
      X(17) = XS(5)-XS(11)-XS(6)
      X(18) = ZT-XS(5)
      X(19) = XS(7)-XS(1)
      X(20) = 0.
      X(21) = XS(9)-XS(3)
      X(22) = 0.
      X(23) = XS(11)-XS(5)
      X(24) = 0.
      T=TF                                                              00
      MB=24
      XJF = JF
      DTB=-(TF-TS)/XJF
      KK=JF+1
      CALL SET(MB,T,X,DTB,SLOPEB,0,.T.,0,0)                             00
      DO 40 JB=1,JF
      KK = KK-1
      CALL STEP(MB,T,X,DTB,SLOPEB,0,.T.,0,0)                            00
      SINU1 = SIN(UT(KK,1))
      COSU1 = COS(UT(KK,1))
      SINU2 = SIN(UT(KK,2))
      COSU2 = COS(UT(KK,2))
      SINV1 = SIN(VT(KK,1))
      COSV1 = COS(VT(KK,1))
      SINV2 = SIN(VT(KK,2))
      COSV2 = COS(VT(KK,2))
      HU(KK,1)=(-X(20)*SINU1+X(22)*COSU1)*TP*COSU2                      00
      HU(KK,2)=(-X(20)*COSU1-X(22)*SINU1)*TP*SINU2+X(24)*TP*COSU2       00
      HV(KK,1)=(-X(14)*SINV1+X(16)*COSV1)*TE*COSV2                      00
      HV(KK,2)=(-X(14)*COSV1-X(16)*SINV1)*TE*SINV2+X(18)*TE*COSV2       00
      IF(ABS(HU(KK,1)).GT.HUMAX1) HUMAX1 = ABS(HU(KK,1))
      IF(ABS(HU(KK,2)).GT.HUMAX2) HUMAX2 = ABS(HU(KK,2))
      IF(ABS(HV(KK,1)).GT.HVMAX1) HVMAX1 = ABS(HV(KK,1))
      IF(ABS(HV(KK,2)).GT.HVMAX2) HVMAX2 = ABS(HV(KK,2))
40    CONTINUE                                                          00
      EPSILON = .5
      HUCHEK=HU(KK,1)**2+HU(KK,2)**2                                    00
      HVCHEK=HV(KK,1)**2+HV(KK,2)**2                                    00
      PRINT*,"CURRENT SAMPLING TIME=",TS
      PRINT 200,HUCHEK,HVCHEK
200   FORMAT ("HUCHEK=",G15.8,2X,"HVCHEK=",G15.8/)
      DO 50 JK=1,JF
      DUT(JK,1)=ABS(HU(JK,1)*DU1MAX/HUMAX1)
      DUT(JK,2)=ABS(HU(JK,2)*DU2MAX/HUMAX2)
      DVT(JK,1)=ABS(HV(JK,1)*DV1MAX/HVMAX1)
      DVT(JK,2)=ABS(HV(JK,2)*DV2MAX/HVMAX2)
      IF(HU(JK,1).GT.0.) DUT(JK,1) = -DUT(JK,1)
      IF(HU(JK,2).GT.0.) DUT(JK,2) = -DUT(JK,2)
```

```
      IF(HV(JK,1).LT.0.) DVT(JK,1) = -DVT(JK,1)
      IF(HV(JK,2).LT.0.) DVT(JK,2) = -DVT(JK,2)
      UT(JK,1) = UT(JK,1)+DUT(JK,1)
      UT(JK,2) = UT(JK,2)+DUT(JK,2)
      VT(JK,1) = VT(JK,1)+DVT(JK,1)
      VT(JK,2) = VT(JK,2)+DVT(JK,2)
      IF(HUCHEK.LE.EPSILON.AND.HVCHEK.LE.EPSILON) GO TO 100            00
      IF(KOUNT.GE.4 ) GO TO 100
50    CONTINUE
      GO TO 5
100   TSLAST = TS
      TS = TS+DEL
      RETURN
      END
      SUBROUTINE F(T,Y,P)
      DIMENSION Y(12),P(12)
      COMMON/BLOK/TP,TE,TF,TS,TSLAST,DEL,U(2),V(2),VNONOPT(2),
     CUT(150,2),VT(150,2),KK,JF,PI,XT,YT,ZT,USTAR(150,2),VSTAR(150,2)
      P(1)  = Y(2)
      P(2)  = TE*COS(VNONOPT(2))*COS(VNONOPT(1))
      P(3)  = Y(4)
      P(4)  = TE*COS(VNONOPT(2))*SIN(VNONOPT(1))
      P(5)  = Y(6)
      P(6)  = TE*SIN(VNONOPT(2))
      P(7)  = Y(8)
      P(8)  = TP*COS(U(2))*COS(U(1))
      P(9)  = Y(10)
      P(10)  = TP*COS(U(2))*SIN(U(1))
      P(11)  = Y(12)
      P(12)  = TP*SIN(U(2))
      END
      SUBROUTINE SLOPEF(MF,T,S,DS)
      DIMENSION S(MF),DS(MF)
      COMMON/BLOK/TP,TE,TF,TS,TSLAST,DEL,U(2),V(2),VNONOPT(2),
     CUT(150,2),VT(150,2),KK,JF,PI,XT,YT,ZT,USTAR(150,2),VSTAR(150,2)
      DS(1) = S(2)
      DS(2) = TE*COS(VT(JF,2))*COS(VT(JF,1))
      DS(3) = S(4)
      DS(4) = TE*COS(VT(JF,2))*SIN(VT(JF,1))
      DS(5) = S(6)
      DS(6) = TE*SIN(VT(JF,2))
      DS(7) = S(8)
      DS(8) = TP*COS(UT(JF,2))*COS(UT(JF,1))
      DS(9) = S(10)
      DS(10) = TP*COS(UT(JF,2))*SIN(UT(JF,1))
      DS(11) = S(12)
      DS(12) = TP*SIN(UT(JF,2))
      END
      SUBROUTINE SLOPEB(MB,T,X,DX)
      DIMENSION X(MB),DX(MB)
      COMMON/BLOK/TP,TE,TF,TS,TSLAST,DEL,U(2),V(2),VNONOPT(2),
     CUT(150,2),VT(150,2),KK,JF,PI,XT,YT,ZT,USTAR(150,2),VSTAR(150,2)
      DX(1) = X(2)
      DX(2) = TE*COS(VT(KK,2))*COS(VT(KK,1))
      DX(3) = X(4)
      DX(4) = TE*COS(VT(KK,2))*SIN(VT(KK,1))
      DX(5) = X(6)
      DX(6) = TE*SIN(VT(KK,2))
```

```
      DX(7)  = X(8)
      DX(8 )  = TP*COS(UT(KK,2))*COS(UT(KK,1))
      DX(9)  = X(10)
      DX(10)  = TP*COS(UT(KK,2))*SIN(UT(KK,1))
      DX(11)  = X(12)
      DX(12)  = TP*SIN(UT(KK,2))
      DX(13)  = 0.
      DX(14)  = -X(13)
      DX(15)  = 0.
      DX(16)  = -X(15)
      DX(17)  = 0.
      DX(18)  = -X(17)
      DX(19)  = 0.
      DX(20)  = -X(19)
      DX(21)  = 0.
      DX(22)  = -X(21)
      DX(23)  = 0.
      DX(24)  = -X(23)
      RETURN
      END
      SUBROUTINE CUPDATE
      COMMON/BLOK/TP,TE,TF,TS,TSLAST,DEL,U(2),V(2),VNONOPT(2),
     CUT(150,2),VT(150,2),KK,JF,PI,XT,YT,ZT,USTAR(150,2),VSTAR(150,2)
      KK = 1
      U(1) = UT(KK,1)
      U(2) = UT(KK,2)
      V(1) = VT(KK,1)
      V(2) = VT(KK,2)
      RETURN
      END
```

Appendix D

Program Listing: A Closed-Loop Gradient-DDP Control Update Algorithm (Not Real-Time)

```
BRR,T70,STCSP.            T760203,BACON,BOX 4087
FTN.
ATTACH,A,AFITSUBROUTINES,ID=AFIT,SN=AFIT.
LIBRARY,A.
LGO.
      PROGRAM GAME2(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
      DIMENSION Y(12),P(12)
      COMMON /BLOK/TP,TE,TF,TS,TSLAST,DEL,U(2),V(2),VNONOPT(2),
     CUT(150,2),VT(150,2),KK,JF,PI,XT,YT,ZT,USTAR(150,2),VSTAR(150,2)
C     THIS PROGRAM USES A COMBINED GRADIENT DDP METHOD
C     (NOT REAL TIME) WHERE GRAD PROVIDES DDP WITH
C     THE NOMINAL CONTROL HISTORIES
      TP = .3
      TE = .1
      PI = ACOS(-1.)
C     SET STATE VALUES AT T=0.
      DO 5 I=1,12
5     Y(I) = 0.
      Y(2) = .57735
      Y(4) = .57735
      Y(5) = 20.
      Y(6) = -.57735
      Y(7) = 20.
      Y(9) = 20.
C     GUESS THE FINAL TIME.
      TF =10.51111
      DT = TF/128.
      XT = 20.
      YT = 20.
      ZT = 1.
C     DEFINE P'S SAMPLING INTERVAL, DEL, TO BE
      DEL = 1.
C     SET E'S NONOPTIMAL CONTROL SEQUENCE TO BE
      VNONOPT(1) = 0.
      VNONOPT(2) = 0.
C     THE INITIAL NOMINAL CONTROLS ARE
      U(1) = -135.*PI/180.
      U(2) = 35.26*PI/180.
      V(1) = 45.*PI/180.
      V(2) = -35.26*PI/180.
C     THE NOMINAL CONTROL HISTORIES FOR USE IN GRAD   THE 1ST TIME ARE
      DO 10 I=1,150
      UT(I,1)= U(1)
      UT(I,2) = U(2)
      VT(I,1) = V(1)
10    VT(I,2) = V(2)
      TS = DEL
      T = 0.
      N = 12
      U1DEG = U(1)*180./PI
      U2DEG = U(2)*180./PI
      V1DEG = V(1)*180./PI
      V2DEG = V(2)*180./PI
      PRINT 100,T,(Y(I),I=1,12)
      PRINT 200,U1DEG,U2DEG,V1DEG,V2DEG
C     INTEGRATE THE STATE EQUATIONS FORWARD IN TIME
      CALL F(T,Y,P)
```

```
20    CALL RKDES(T,Y,M,DT)
      U1DEG = U(1)*180./PI
      U2DEG = U(2)*180./PI
      V1DEG = V(1)*180./PI
      V2DEG = V(2)*180./PI
      PRINT 100,T,(Y(I),I=1,12)
      PRINT 200,U1DEG,U2DEG,V1DEG,V2DEG
100   FORMAT (/"TIME=",F10.5,2X,"E'S STATE=",6(F9.5,1X),
     C/,17X,"P'S STATE=",6(F9.5,1X))
200   FORMAT ("U=",2(F10.5,2X),"V=",2(F10.5,2X))
      TSCKL= TS-.05
      TSCKH= TS+.05
      IF(T .GT.0.80*TF) DEL = .1
      IF(T.GE.TSCKL.AND.T .LE.TSCKH) CALL GRAD(Y)
      IF(T.GE.TSCKL.AND.T .LE.TSCKH.AND.TS.GT.0.80*TF) CALL DDP(Y)
      IF(T.GE.TSCKL.AND.T .LE.TSCKH) CALL CUPDATE
      EPRANGE=SQRT((Y(1)-Y(7))**2+(Y(3)-Y(9))**2+(Y(5)-Y(11))**2)
      XJ=Y(2)*(XT-Y(1))+Y(4)*(YT-Y(3))+Y(6)*(ZT-Y(5))+.5*EPRANGE**2
      ETRANGE=SQRT((Y(1)-XT)**2+(Y(3)-YT)**2+(Y(5)-ZT)**2)
      PRINT 300,XJ,EPRANGE,ETRANGE
300   FORMAT ("COST=",F10.5,2X,"EPRANGE=",F10.5,2X,"ETRANGE=",F10.5)
      IF(T.LE.TF) GO TO 20
      END
      SUBROUTINE GRAD(Y)
      DIMENSION Y(12),XS(12), X(25),HU(150,2),HV(150,2),DUT(150,2),
     CDVT(150,2)
      COMMON /BLOK/TP,TE,TF,TS,TSLAST,DEL,U(2),V(2),VNONOPT(2),
     CUT(150,2),VT(150,2),KK,JF,PI,XT,YT,ZT,USTAR(150,2),VSTAR(150,2)
      EXTERNAL SLOPEF,SLOPEB
      KOUNT= 0
      XJDOTL = 0.0
      DU1MAX = 3.0*PI/180.
      DU2MAX = 3.0*PI/180.
      DV1MAX = 3.0*PI/180.
      DV2MAX = 3.0*PI/180.
      HUMAX1 = 1.
      HUMAX2 = 1.
      HVMAX1 = 1.
      HVMAX2 = 1.
5     KOUNT= KOUNT+1
      DO 10 I=1,12
10    XS(I)= Y(I)
      T=TS
      MF=12
      DTF=(TF-TS)/64.
      CALL SET(MF,T,XS,DTF,SLOPEF,D,.T.,D,D)
      IF=0
20    IF=IF+1
      JF=IF
      CALL STEP(MF,T,XS,DTF,SLOPEF,D,.T.,D,D)
      DDOT=(XS(1)-XS(7))*(XS(2)-XS(8))+(XS(3)-XS(9))*(XS(4)-XS(10))+
     C(XS(5)-XS(11))*(XS(6)-XS(12))
      FDOT =-XS(2)**2+(XT-XS(1))*TE*COS(VT(JF,2))*COS(VT(JF,1))
     C-XS(4)**2+(YT-XS(3))*TE*COS(VT(JF,2))*SIN(VT(JF,1))
     C-XS(6)**2+(ZT-XS(5))*TE*SIN(VT(JF,2))
      XJDOT = DDOT + FDOT
      IF(XJDOTL*XJDOT.LT.0.0) GO TO 25
      IF(JF.GE.150) GO TO 25
```

```
      XJDOT_ = XJDOT
      GO TO 20                                                          00
25    TF = T
      PRINT 1000,XJDOT,JF,TF
1000  FORMAT ("XJDOT=",G15.8,"JF=",I8,"TF=",G15.8)
      DO 30 KI=1,12
30    X(KI) = XS(KI)
      X(13) = XS(1)-XS(7)-XS(2)
      X(14) = XT-XS(1)
      X(15) = XS(3)-XS(9)-XS(4)
      X(16) = YT-XS(3)
      X(17) = XS(5)-XS(11)-XS(6)
      X(18) = ZT-XS(5)
      X(19) = XS(7)-XS(1)
      X(20) = 0.
      X(21) = XS(9)-XS(3)
      X(22) = 0.
      X(23) = XS(11)-XS(5)
      X(24) = 0.
      X(25) = 0.
      X(26) = 0.
      T=TF                                                              00
      MB=26
      XJF = JF
      DTB=-(TF-TS)/XJF
      KK=JF+1
      CALL SET(MB,T,X,DTB,SLOPEB,D,.T.,D,D)                             00
      DO 40 JB=1,JF
      KK = KK-1
      CALL STEP(MB,T,X,DTB,SLOPEB,D,.T.,D,D)                            00
      SINU1 = SIN(UT(KK,1))
      COSU1 = COS(UT(KK,1))
      SINU2 = SIN(UT(KK,2))
      COSU2 = COS(UT(KK,2))
      SINV1 = SIN(VT(KK,1))
      COSV1 = COS(VT(KK,1))
      SINV2 = SIN(VT(KK,2))
      COSV2 = COS(VT(KK,2))
      HU(KK,1)=(-X(20)*SINU1+X(22)*COSU1)*TP*COSU2                      00
      HU(KK,2)=(-X(20)*COSU1-X(22)*SINU1)*TP*SINU2+X(24)*TP*COSU2       00
      HV(KK,1)=(-X(14)*SINV1+X(16)*COSV1)*TE*COSV2                      00
      HV(KK,2)=(-X(14)*COSV1-X(16)*SINV1)*TE*SINV2+X(18)*TE*COSV2       00
      IF(ABS(HU(KK,1)).GT.HUMAX1) HUMAX1 = ABS(HU(KK,1))
      IF(ABS(HU(KK,2)).GT.HUMAX2) HUMAX2 = ABS(HU(KK,2))
      IF(ABS(HV(KK,1)).GT.HVMAX1) HVMAX1 = ABS(HV(KK,1))
      IF(ABS(HV(KK,2)).GT.HVMAX2) HVMAX2 = ABS(HV(KK,2))
40    CONTINUE                                                          00
      EPSILON = .1
      HUCHEK=HU(KK,1)**2+HU(KK,2)**2                                    00
      HVCHEK=HV(KK,1)**2+HV(KK,2)**2                                    00
      PRINT*,"CURRENT SAMPLING TIME=",TS
      PRINT 200,HUCHEK,HVCHEK
200   FORMAT ("HUCHEK=",G15.8,2X,"HVCHEK=",G15.8/)
      DO 50 JK=1,JF
      DUT(JK,1)=ABS(HU(JK,1)*DU1MAX/HUMAX1)
      DUT(JK,2)=ABS(HU(JK,2)*DU2MAX/HUMAX2)
      DVT(JK,1)=ABS(HV(JK,1)*DV1MAX/HVMAX1)
      DVT(JK,2)=ABS(HV(JK,2)*DV2MAX/HVMAX2)
```

```
      IF(HU(JK,1).GT.0.) DUT(JK,1) = -DUT(JK,1)
      IF(HU(JK,2).GT.0.) DUT(JK,2) = -DUT(JK,2)
      IF(HV(JK,1).LT.0.) DVT(JK,1) = -DVT(JK,1)
      IF(HV(JK,2).LT.0.) DVT(JK,2) = -DVT(JK,2)
      UT(JK, 1) = UT(JK,1)+DUT(JK,1)
      UT(JK, 2) = UT(JK,2)+DUT(JK,2)
      VT(JK, 1) = VT(JK,1)+DVT(JK,1)
      VT(JK, 2) = VT(JK,2)+DVT(JK,2)
      IF(HUCHEK.LE.EPSILON.AND.HVCHEK.LE.EPSILON) GO TO 100               00
      IF(KOUNT.GE.5 ) GO TO 100
50    CONTINUE
      GO TO 5
100   TSLAST = TS
      TS = TS+DEL
      RETURN
      END
      SUBROUTINE DOP(Y)
      DIMENSION Y(12),XS(12),X(26)
      COMMON /BLOK/TP,TE,TF,TS,TSLAST,DEL,U(2),V(2),VNONOPT(2),
     CUT(150,2),VT(150,2),KK,JF,PI,XT,YT,ZT,USTAR(150,2),VSTAR(150,2)
      EXTERNAL SLOPEF,SLOPEB
      KOUNT= 0
      XJL = 0.
      XJDOTL = 0.0
5     KOUNT= KOUNT+1
      DO 10 I=1,12
10    XS(I)=Y(I)
      T=TS
      MF=12
      DTF=(TF-TS)/64.
      CALL SET(MF,T,XS,DTF,SLOPEF,D,.T.,D,D)
      IF=0
20    IF=IF+1
      JF=IF
      CALL STEP(MF,T,XS,DTF,SLOPEF,D,.T.,D,D)
      DDOT=(XS(1)-XS(7))*(XS(2)-XS(8))+(XS(3)-XS(9))*(XS(4)-XS(10))+
     C(XS(5)-XS(11))*(XS(6)-XS(12))
      FDOT =-XS(2)**2+(XT-XS(1))*TE*COS(VT(JF,2))*COS(VT(JF,1))
     C-XS(4)**2+(YT-XS(3))*TE*COS(VT(JF,2))*SIN(VT(JF,1))
     C-XS(6)**2+(ZT-XS(5))*TE*SIN(VT(JF,2))
      XJDOT = DDOT + FDOT
      IF(XJDOTL*XJDOT.LT.0.0) GO TO 25
      IF(JF.GE.65 ) GO TO 25
      XJDOTL = XJDOT
      GO TO 20
25    TF = T
      EPRSQD = (XS(1)-XS(7))**2+(XS(3)-XS(9))**2+
     C(XS(5)-XS(11))**2
      XJ = XS(2)*(XT-XS(1))+XS(4)*(YT-XS(3))+
     CXS(6)*(ZT-XS(5))+.5*EPRSQD
      DJATTF = XJ-XJL
      PRINT 1000,XJDOT,JF,TF
1000  FORMAT("XJDOT=",G15.8,"JF=",I8,"TF=",G15.8)
      DO 30 KI=1,12
30    X(KI) = XS(KI)
      X(13) = XS(1)-XS(7)-XS(2)
      X(14) = XT-XS(1)
      X(15) = XS(3)-XS(9)-XS(4)
```

```
      X(16)  = YT-XS(3)
      X(17)  = XS(5)-XS(11)-XS(6)
      X(18)  = ZT-XS(5)
      X(19)  = XS(7)-XS(1)
      X(20)  = 0.
      X(21)  = XS(9)-XS(3)
      X(22)  = 0.
      X(23)  = XS(11)-XS(5)
      X(24)  = 0.
      X(25)  = 0.
      X(26)  =0.
      T=TF
      MB=26
      XJF = JF
      DTB=-(TF-TS)/XJF
      KK=JF+1
      CALL SET(MB,T,X,DTB,SLOPEB,D,.T.,D,D)
      DO 40  JB=1,JF
      KK = KK-1
      CALL STEP(MB,T,X,DTB, SLOPEB,D,.T.,D,D)
40    CONTINUE
      DO 80  JK=1,JF
      UT(JK,1)  = USTAR(JK,1)
      UT(JK,2)  = USTAR(JK,2)
      VT(JK,1)  = VSTAR(JK,1)
      VT(JK,2)  = VSTAR(JK,2)
80    CONTINUE
      ATS  = X(25)+X(26)
      PRINT  200,X(25),X(26) ,ATS,DJATTF
200   FORMAT ("AE  AT  TS=",G15.8,2X,"AP  AT  TS=",G15.8,2X,"A=",G15.8/
     C"DJ  AT  TF=",G15.8/)
      PRINT  300,KOUNT
300   FORMAT ("KOUNT=",I5)
      EPSILON = .01
      IF(ABS(ATS).LE.EPSILON)  GO TO 100
      IF(KOUNT.GE.5 )  GO TO 100
      GO TO 5
100   RETURN
      END
      SUBROUTINE  F(T,Y,P)
      DIMENSION Y(12),P(12)
      COMMON /BLOK/TP,TE,TF,TS,TSLAST,DEL,U(2),V(2),VNONOPT(2),
     CUT(150,2),VT(150,2),KK,JF,PI,XT,YT,ZT,USTAR(150,2),VSTAR(150,2)
      P(1)   = Y(2)
      P(2)   = TE*COS(VNONOPT(2))*COS(VNONOPT(1))
      P(3)   = Y(4)
      P(4)   = TE*COS(VNONOPT(2))*SIN(VNONOPT(1))
      P(5)   = Y(6)
      P(6)   = TE*SIN(VNONOPT(2))
      P(7)   = Y(8)
      P(8)   = TP*COS(U(2))*COS(U(1))
      P(9)   = Y(10)
      P(10)   = TP*COS(U(2))*SIN(U(1))
      P(11)   = Y(12)
      P(12)   = TP*SIN(U(2))
      END
      SUBROUTINE  SLOPEF(MF,T,S,DS)
      DIMENSION S(MF),DS(MF)
```

```
      COMMON /BLOK/TP,TE,TF,TS,TSLAST,DEL,U(2),V(2),VNONOPT(2),
     CUT(150,2),VT(150,2),KK,JF,PI,XT,YT,ZT,USTAR(150,2),VSTAR(150,2)
      DS(1)  = S(2)
      DS(2)  = TE*COS(VT(JF,2))*COS(VT(JF,1))
      DS(3)  = S(4)
      DS(4)  = TE*COS(VT(JF,2))*SIN(VT(JF,1))
      DS(5)  = S(6)
      DS(6)  = TE*SIN(VT(JF,2))
      DS(7)  = S(8)
      DS(8)  = TP*COS(UT(JF,2))*COS(UT(JF,1))
      DS(9)  = S(10)
      DS(10)  = TP*COS(UT(JF,2))*SIN(UT(JF,1))
      DS(11)  = S(12)
      DS(12)  = TP*SIN(UT(JF,2))
      END
      SUBROUTINE SLOPER(MB,T,X,DX)
      DIMENSION X(MB),DX(MB)
      COMMON /BLOK/TP,TE,TF,TS,TSLAST,DEL,U(2),V(2),VNONOPT(2),
     CUT(150,2),VT(150,2),KK,JF,PI,XT,YT,ZT,USTAR(150,2),VSTAR(150,2)
      SQRRTE1 = SQRT(X(14)**2+X(16)**2)
      SQRRTE2 = SQRT(SQRRTE1**2+X(18)**2)
      SINV1      = X(16)/SQRRTE1
      COSV1      = X(14)/SQRRTE1
      SINV2      = X(18)/SQRRTE2
      COSV2      = SQRRTE1/SQRRTE2
      VSTAR(KK,1)  = ATAN2(SINV1,COSV1)
      VSTAR(KK,2)  = ATAN2(SINV2,COSV2)
      SQRRTP1 = -SQRT(X(20)**2+X(22)**2)
      SQRRTP2 = -SQRT(SQRRTP1**2+X(24)**2)
      IF(SQRRTP1.EQ.0.0) GO TO 1
      SINU1      = X(22)/SQRRTP1
      COSU1      = X(20)/SQRRTP1
      SINU2      = X(24)/SQRRTP2
      COSU2      = SQRRTP1/SQRRTP2
      USTAR(KK,1)  = ATAN2(SINU1,COSU1)
      USTAR(KK,2)  = ATAN2(SINU2,COSU2)
      GO TO 5
 1    USTAR(KK,1)  = -135.*PI/180.
      USTAR(KK,2)  = 35.26*PI/180.
 5    DX(1)  = X(2)
      DX(2)  = TE*COS(VT(KK,2))*COS(VT(KK,1))
      DX(3)  = X(4)
      DX(4)  = TE*COS(VT(KK,2))*SIN(VT(KK,1))
      DX(5)  = X(6)
      DX(6)  = TE*SIN(VT(KK,2))
      DX(7)  = X(8)
      DX(8 )  = TP*COS(UT(KK,2))*COS(UT(KK,1))
      DX(9)  = X(10)
      DX(10)  = TP*COS(UT(KK,2))*SIN(UT(KK,1))
      DX(11)  = X(12)
      DX(12)  = TP*SIN(UT(KK,2))
      DX(13)  = 0.
      DX(14)  = -X(13)
      DX(15)  = 0.
      DX(16)  = -X(15)
      DX(17)  = 0.
      DX(18)  = -X(17)
      DX(19)  = 0.
```

```
      DX(20)  =  -X(19)
      DX(21)  =  0.
      DX(22)  =  -X(21)
      DX(23)  =  0.
      DX(24)  =  -X(23)
      I=12
      DO 10 L=1,6
      I=I+1
10    HOLDE  =  DX(L)*X(I)
      DO 15 L=7,12
      I = I + 1
15    HOLDP  =  DX(L)*X(I)
      HNEWE= X(13)*X(2)+X(14)*TE*COS(VSTAR(KK,2))*COS(VSTAR(KK,1))+
     CX(15)*X(4)+X(16)*TE*COS(VSTAR(KK,2))*SIN(VSTAR(KK,1))+
     CX(17)*X(6)+X(18)*TE*SIN(VSTAR(KK,2))
      HNEWP= X(19)*X(8)+X(20)*TP*COS(USTAR(KK,2))*COS(USTAR(KK,1))+
     CX(21)*X(10)+X(22)*TP*COS(USTAR(KK,2))*SIN(USTAR(KK,1))+
     CX(23)*X(12)+X(24)*TP*SIN(USTAR(KK,2))
      DX(25)  =  HOLDE-HNEWE
      DX(26)  =  HOLDP-HNEWP
      RETURN
      END
      SUBROUTINE CUPDATE
      COMMON/BLOK/TP,TE,TF,TS,TSLAST,DEL,U(2),V(2),VNONOPT(2),
     CUT(150,2),VT(150,2),KK,JF,PI,XT,YT,ZT,USTAR(150,2),VSTAR(150,2)
      KK = 1
      U(1)  =  UT(KK,1)
      U(2)  =  UT(KK,2)
      V(1)  =  VT(KK,1)
      V(2)  =  VT(KK,2)
      RETURN
      END
```

Vita

Robert R. Bacon was born on 1 June 1949 in Nyack, New York, and graduated from Nyack High School in 1967. He attended Virginia Polytechnic Institute and State University from September 1967 until June 1971 and earned a Bachelor of Science degree in Aerospace Engineering. While at V.P.I. he was a trombonist with the Virginia Tech Regimental Band. Upon graduation, he was commissioned through the USAF ROTC program and was assigned to attend the Communications Officer School at Keesler AFB. Upon completion of training in February 1972 he was assigned to the Distant Early Warning (D.E.W.) Line System Office of the Aerospace Defense Command (A.D.C.) and served as a staff officer in the Telecommunications Branch of the D.E.W. Office. In June, 1975, he entered the Graduate Astronautical Engineering program at the Air Force Institute of Technology.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER GA/MC/76D-2	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) CLOSED-LOOP CONTROLS FOR DIFFERENTIAL GAMES USING A GRADIENT AND A DIFFERENTIAL DYNAMIC PROGRAMMING METHOD		5. TYPE OF REPORT & PERIOD COVERED M.S. Thesis 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Robert R. Bacon Captain, USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT-EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE December, 1976 13. NUMBER OF PAGES 70
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; IAW AFR 190-17

JERRAL F. GUESS, Captain, USAF
Director of Information

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Differential Games
Pursuit-Evasion
Trajectory Optimization
Zero-Sum Games
Optimal Strategies
Closed-Loop Controls

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis investigates the use of a gradient method and a combined gradient-differential dynamic programming (DDP) method to generate closed-loop controls for intercept problems formulated as differential games. The gradient method is applied to a planar motion pursuit-evasion game. The trajectory obtained compares favorably with that obtained using analytic expressions for closed-loop controls. The gradient method is applied to a three dimensional

next page

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

cont

interceptor-penetrator game with simplified dynamics on a real-time basis. A combined gradient-DDP algorithm is applied to this problem but not on a real-time basis. The DDP portion of this combined control law was found to be unstable. The results obtained indicate that a gradient based scheme, because of its numerical stability and ability to rapidly converge to the vicinity of the optimum, may be used to generate an effective near-optimal closed-loop control law for some problems.